```
000001  ############################################################################
000002  ### FILE: ADSP.p
000003  ############################################################################
000004
000005
000006  {
000007  Created: Sunday, September 15, 1991 at 9:36 PM
000008   ADSP.p
000009   Pascal Interface to the Macintosh Libraries
000010
000011    Copyright Apple Computer, Inc. 1986-1991
000012    All rights reserved
000013  }
000014
000015
000016  {$IFC UNDEFINED UsingIncludes}
000017  {$SETC UsingIncludes := 0}
000018  {$ENDC}
000019
000020  {$IFC NOT UsingIncludes}
000021   UNIT ADSP;
000022   INTERFACE
000023  {$ENDC}
000024
000025  {$IFC UNDEFINED UsingADSP}
000026  {$SETC UsingADSP := 1}
000027
000028  {$I+}
000029  {$SETC ADSPIncludes := UsingIncludes}
000030  {$SETC UsingIncludes := 1}
000031  {$IFC UNDEFINED UsingAppleTalk}
000032  {$I $$Shell(PInterfaces)AppleTalk.p}
000033  {$ENDC}
000034  {$SETC UsingIncludes := ADSPIncludes}
000035
000036  CONST
000037
000038  { driver control ioResults }
000039  errRefNum = -1280;                      { bad connection refNum }
000040  errAborted = -1279;                     { control call was aborted }
000041  errState = -1278;                       { bad connection state for this operation }
000042  errOpening = -1277;                     { open connection request failed }
000043  errAttention = -1276;                   { attention message too long }
000044  errFwdReset = -1275;                    { read terminated by forward reset }
000045  errDSPQueueSize = -1274;                { DSP Read/Write Queue Too small }
000046  errOpenDenied = -1273;                  { open connection request was denied }
000047
000048  {driver control csCodes}
000049  dspInit = 255;                          { create a new connection end }
000050  dspRemove = 254;                        { remove a connection end }
000051  dspOpen = 253;                          { open a connection }
000052  dspClose = 252;                         { close a connection }
000053  dspCLInit = 251;                        { create a connection listener }
000054  dspCLRemove = 250;                      { remove a connection listener }
000055  dspCLListen = 249;                      { post a listener request }
000056  dspCLDeny = 248;                        { deny an open connection request }
```

```
000057  dspStatus = 247;                            { get status of connection end }
000058  dspRead = 246;                              { read data from the connection }
000059  dspWrite = 245;                             { write data on the connection }
000060  dspAttention = 244;                         { send an attention message }
000061  dspOptions = 243;                           { set connection end options }
000062  dspReset = 242;                             { forward reset the connection }
000063  dspNewCID = 241;                            { generate a cid for a connection end }
000064
000065  { connection opening modes }
000066  ocRequest = 1;                              { request a connection with remote }
000067  ocPassive = 2;                              { wait for a connection request from remote }
000068  ocAccept = 3;                               { accept request as delivered by listener }
000069  ocEstablish = 4;                            { consider connection to be open }
000070
000071  { connection end states }
000072  sListening = 1;                             { for connection listeners }
000073  sPassive = 2;                               { waiting for a connection request from remote }
000074  sOpening = 3;                               { requesting a connection with remote }
000075  sOpen = 4;                                  { connection is open }
000076  sClosing = 5;                               { connection is being torn down }
000077  sClosed = 6;                                { connection end state is closed }
000078
000079  { client event flags }
000080  eClosed = $80;                              { received connection closed advice }
000081  eTearDown = $40;                            { connection closed due to broken connection }
000082  eAttention = $20;                           { received attention message }
000083  eFwdReset = $10;                            { received forward reset advice }
000084
000085  { miscellaneous constants }
000086  attnBufSize = 570;                          { size of client attention buffer }
000087  minDSPQueueSize = 100;                      { Minimum size of receive or send Queue }
000088
000089  TYPE
000090  { connection control block }
000091  TPCCB = ^TRCCB;
000092  TRCCB = PACKED RECORD
000093   ccbLink: TPCCB;                            { link to next ccb }
000094   refNum: INTEGER;                           { user reference number }
000095   state: INTEGER;                            { state of the connection end }
000096   userFlags: Byte;                           { flags for unsolicited connection events }
000097   localSocket: Byte;                         { socket number of this connection end }
000098   remoteAddress: AddrBlock;                  { internet address of remote end }
000099   attnCode: INTEGER;                         { attention code received }
000100   attnSize: INTEGER;                         { size of received attention data }
000101   attnPtr: Ptr;                              { ptr to received attention data }
000102   reserved: PACKED ARRAY [1..220] OF Byte;   { adsp internal use }
000103   END;
000104
000105  { ADSP CntrlParam ioQElement , driver control call parameter block}
000106  DSPPBPtr = ^DSPParamBlock;
000107  DSPParamBlock = PACKED RECORD
000108   qLink: QElemPtr;
000109   qType: INTEGER;
000110   ioTrap: INTEGER;
000111   ioCmdAddr: Ptr;
000112   ioCompletion: ProcPtr;
```

```
000113   ioResult: OSErr;
000114   ioNamePtr: StringPtr;
000115   ioVRefNum: INTEGER;
000116   ioCRefNum: INTEGER;                    { adsp driver refNum }
000117   csCode: INTEGER;                       { adsp driver control code }
000118   qStatus: LONGINT;                      { adsp internal use }
000119   ccbRefNum: INTEGER;                    { refnum of ccb }
000120   CASE INTEGER OF
000121     dspInit,dspCLInit:
000122    (ccbPtr: TPCCB;                       {pointer to connection control block}
000123     userRoutine: ProcPtr;                {client routine to call on event}
000124     sendQSize: INTEGER;                  {size of send queue (0..64K bytes)}
000125     sendQueue:  Ptr;                     {client passed send queue buffer}
000126     recvQSize: INTEGER;                  {size of receive queue (0..64K bytes)}
000127     recvQueue: Ptr;                      {client passed receive queue buffer}
000128     attnPtr: Ptr;                        {client passed receive attention buffer}
000129     localSocket: Byte;                   {local socket number}
000130     filler1: Byte;                       {filler for proper byte alignment}
000131      );
000132      dspOpen,dspCLListen,dspCLDeny:
000133     (localCID: INTEGER;                   {local connection id}
000134      remoteCID: INTEGER;                  {remote connection id}
000135      remoteAddress: AddrBlock;            {address of remote end}
000136      filterAddress: AddrBlock;            {address filter}
000137      sendSeq: LONGINT;                    {local send sequence number}
000138      sendWindow: INTEGER;                 {send window size}
000139      recvSeq: LONGINT;                    {receive sequence number}
000140      attnSendSeq: LONGINT;                {attention send sequence number}
000141      attnRecvSeq: LONGINT;                {attention receive sequence number}
000142      ocMode: Byte;                        {open connection mode}
000143      ocInterval: Byte;                    {open connection request retry interval}
000144      ocMaximum: Byte;                     {open connection request retry maximum}
000145      filler2: Byte;                       {filler for proper byte alignment}
000146       );
000147       dspClose,dspRemove:
000148      (abort: Byte;                         {abort connection immediately if non-zero}
000149       filler3: Byte;                       {filler for proper byte alignment}
000150        );
000151        dspStatus:
000152       (statusCCB: TPCCB;                   {pointer to ccb}
000153        sendQPending: INTEGER;              {pending bytes in send queue}
000154        sendQFree: INTEGER;                 {available buffer space in send queue}
000155        recvQPending: INTEGER;              {pending bytes in receive queue}
000156        recvQFree: INTEGER;                 {available buffer space in receive queue}
000157         );
000158         dspRead,dspWrite:
000159        (reqCount: INTEGER;                 {requested number of bytes}
000160         actCount: INTEGER;                 {actual number of bytes}
000161         dataPtr: Ptr;                      {pointer to data buffer}
000162         eom: Byte;                         {indicates logical end of message}
000163         flush: Byte;                       {send data now}
000164          );
000165          dspAttention:
000166         (attnCode: INTEGER;                 {client attention code}
000167          attnSize: INTEGER;                 {size of attention data}
000168          attnData: Ptr;                     {pointer to attention data}
```

```
000169          attnInterval: Byte;                {retransmit timer in 10-tick intervals}
000170          filler4: Byte;                     {filler for proper byte alignment}
000171           );
000172           dspOptions:
000173          (sendBlocking: INTEGER;             {quantum for data packets}
000174           sendTimer: Byte;                   {send timer in 10-tick intervals}
000175           rtmtTimer: Byte;                   {retransmit timer in 10-tick intervals}
000176           badSeqMax: Byte;                   {threshold for sending retransmit advice}
000177           useCheckSum: Byte;                 {use ddp packet checksum}
000178            );
000179            dspNewCID:
000180           (newCID: INTEGER;                  {new connection id returned}
000181             );
000182   END;
000183
000184
000185
000186  {$ENDC} { UsingADSP }
000187
000188  {$IFC NOT UsingIncludes}
000189   END.
000190  {$ENDC}
000191
000192
000193  ### END OF FILE ADSP.p
000194
```

```
000195
000196  ############################################################################
000197  ### FILE: AIFF.p
000198  ############################################################################
000199
000200
000201  {
000202  Created: Monday, December 2, 1991 at 5:01 PM
000203   AIFF.p
000204   Pascal Interface to the Macintosh Libraries
000205
000206    Copyright Apple Computer, Inc. 1990-1991
000207    All rights reserved
000208  }
000209
000210
000211  {$IFC UNDEFINED UsingIncludes}
000212  {$SETC UsingIncludes := 0}
000213  {$ENDC}
000214
000215  {$IFC NOT UsingIncludes}
000216   UNIT AIFF;
000217   INTERFACE
000218  {$ENDC}
000219
000220  {$IFC UNDEFINED UsingAIFF}
000221  {$SETC UsingAIFF := 1}
000222
000223  {$I+}
000224  {$SETC AIFFIncludes := UsingIncludes}
000225  {$SETC UsingIncludes := 1}
000226  {$IFC UNDEFINED UsingTypes}
000227  {$I $$Shell(PInterfaces)Types.p}
000228  {$ENDC}
000229  {$SETC UsingIncludes := AIFFIncludes}
000230
000231  CONST
000232  AIFFID = 'AIFF';
000233  AIFCID = 'AIFC';
000234  FormatVersionID = 'FVER';
000235  CommonID = 'COMM';
000236  FORMID = 'FORM';
000237  SoundDataID = 'SSND';
000238  MarkerID = 'MARK';
000239  InstrumentID = 'INST';
000240  MIDIDataID = 'MIDI';
000241  AudioRecordingID = 'AESD';
000242  ApplicationSpecificID = 'APPL';
000243  CommentID = 'COMT';
000244  NameID = 'NAME';
000245  AuthorID = 'AUTH';
000246  CopyrightID = '(c) ';
000247  AnnotationID = 'ANNO';
000248  NoLooping = 0;
000249  ForwardLooping = 1;
000250  ForwardBackwardLooping = 2;
```

```
000251
000252  { AIFF-C Versions }
000253  AIFCVersion1 = $A2805140;
000254
000255  { Compression Names }
000256  NoneName = 'not compressed';
000257  ACE2to1Name = 'ACE 2-to-1';
000258  ACE8to3Name = 'ACE 8-to-3';
000259  MACE3to1Name = 'MACE 3-to-1';
000260  MACE6to1Name ='MACE 6-to-1';
000261
000262  { Compression Types }
000263  NoneType = 'NONE';
000264  ACE2Type = 'ACE2';
000265  ACE8Type = 'ACE8';
000266  MACE3Type = 'MAC3';
000267  MACE6Type = 'MAC6';
000268
000269  TYPE
000270  ID = LONGINT;
000271  MarkerIdType = INTEGER;
000272
000273  ChunkHeader = RECORD
000274   ckID: ID;
000275   ckSize: LONGINT;
000276   END;
000277
000278  ContainerChunk = RECORD
000279   ckID: ID;
000280   ckSize: LONGINT;
000281   formType: ID;
000282   END;
000283
000284  FormatVersionChunkPtr = ^FormatVersionChunk;
000285  FormatVersionChunk = RECORD
000286   ckID: ID;
000287   ckSize: LONGINT;
000288   timestamp: LONGINT;
000289   END;
000290
000291  CommonChunkPtr = ^CommonChunk;
000292  CommonChunk = RECORD
000293   ckID: ID;
000294   ckSize: LONGINT;
000295   numChannels: INTEGER;
000296   numSampleFrames: LONGINT;
000297   sampleSize: INTEGER;
000298   sampleRate: Extended80;
000299   END;
000300
000301  ExtCommonChunkPtr = ^ExtCommonChunk;
000302  ExtCommonChunk = RECORD
000303   ckID: ID;
000304   ckSize: LONGINT;
000305   numChannels: INTEGER;
000306   numSampleFrames: LONGINT;
```

```
000307   sampleSize: INTEGER;
000308   sampleRate: Extended80;
000309   compressionType: ID;
000310   compressionName: PACKED ARRAY [0..0] OF Byte;
000311   END;
000312
000313  SoundDataChunkPtr = ^SoundDataChunk;
000314  SoundDataChunk = RECORD
000315   ckID: ID;
000316   ckSize: LONGINT;
000317   offset: LONGINT;
000318   blockSize: LONGINT;
000319   END;
000320
000321  Marker = RECORD
000322   id: MarkerIdType;
000323   position: LONGINT;
000324   markerName: Str255;
000325   END;
000326
000327  MarkerChunkPtr = ^MarkerChunk;
000328  MarkerChunk = RECORD
000329   ckID: ID;
000330   ckSize: LONGINT;
000331   numMarkers: INTEGER;
000332   Markers: ARRAY [0..0] OF Marker;
000333   END;
000334
000335  AIFFLoop = RECORD
000336   playMode: INTEGER;
000337   beginLoop: MarkerIdType;
000338   endLoop: MarkerIdType;
000339   END;
000340
000341  InstrumentChunkPtr = ^InstrumentChunk;
000342  InstrumentChunk = PACKED RECORD
000343   ckID: ID;
000344   ckSize: LONGINT;
000345   baseFrequency: Byte;
000346   detune: Byte;
000347   lowFrequency: Byte;
000348   highFrequency: Byte;
000349   lowVelocity: Byte;
000350   highVelocity: Byte;
000351   gain: INTEGER;
000352   sustainLoop: AIFFLoop;
000353   releaseLoop: AIFFLoop;
000354   END;
000355
000356  MIDIDataChunkPtr = ^MIDIDataChunk;
000357  MIDIDataChunk = RECORD
000358   ckID: ID;
000359   ckSize: LONGINT;
000360   MIDIdata: PACKED ARRAY [0..0] OF SignedByte;
000361   END;
000362
```

```
000363  AudioRecordingChunkPtr = ^AudioRecordingChunk;
000364  AudioRecordingChunk = RECORD
000365   ckID: ID;
000366   ckSize: LONGINT;
000367   AESChannelStatus: PACKED ARRAY [0..23] OF SignedByte;
000368   END;
000369
000370  ApplicationSpecificChunkPtr = ^ApplicationSpecificChunk;
000371  ApplicationSpecificChunk = RECORD
000372   ckID: ID;
000373   ckSize: LONGINT;
000374   applicationSignature: OSType;
000375   data: PACKED ARRAY [0..0] OF Byte;
000376   END;
000377
000378  Comment = RECORD
000379   timeStamp: LONGINT;
000380   marker: MarkerIdType;
000381   count: INTEGER;
000382   text: PACKED ARRAY [0..0] OF Byte;
000383   END;
000384
000385  CommentsChunkPtr = ^CommentsChunk;
000386  CommentsChunk = RECORD
000387   ckID: ID;
000388   ckSize: LONGINT;
000389   numComments: INTEGER;
000390   comments: ARRAY [0..0] OF Comment;
000391   END;
000392
000393  TextChunkPtr = ^TextChunk;
000394  TextChunk = RECORD
000395   ckID: ID;
000396   ckSize: LONGINT;
000397   text: PACKED ARRAY [0..0] OF Byte;
000398   END;
000399
000400
000401
000402  {$ENDC} { UsingAIFF }
000403
000404  {$IFC NOT UsingIncludes}
000405   END.
000406  {$ENDC}
000407
000408
000409  ### END OF FILE AIFF.p
000410
```

```
000411
000412  ##########################################################################
000413  ### FILE: Aliases.p
000414  ##########################################################################
000415
000416  {
000417  Created: Monday, January 28, 1991 at 1:26 PM
000418      Aliases.p
000419      Pascal Interface to the Macintosh Libraries
000420
000421          Copyright Apple Computer, Inc.    1989-1990
000422          All rights reserved
000423  }
000424
000425
000426  {$IFC UNDEFINED UsingIncludes}
000427  {$SETC UsingIncludes := 0}
000428  {$ENDC}
000429
000430  {$IFC NOT UsingIncludes}
000431      UNIT Aliases;
000432      INTERFACE
000433  {$ENDC}
000434
000435  {$IFC UNDEFINED UsingAliases}
000436  {$SETC UsingAliases := 1}
000437
000438  {$I+}
000439  {$SETC AliasesIncludes := UsingIncludes}
000440  {$SETC UsingIncludes := 1}
000441  {$IFC UNDEFINED UsingTypes}
000442  {$I $$Shell(PInterfaces)Types.p}
000443  {$ENDC}
000444  {$IFC UNDEFINED UsingDialogs}
000445  {$I $$Shell(PInterfaces)Dialogs.p}
000446  {$ENDC}
000447  {$IFC UNDEFINED UsingAppleTalk}
000448  {$I $$Shell(PInterfaces)AppleTalk.p}
000449  {$ENDC}
000450  {$IFC UNDEFINED UsingFiles}
000451  {$I $$Shell(PInterfaces)Files.p}
000452  {$ENDC}
000453  {$SETC UsingIncludes := AliasesIncludes}
000454
000455  CONST
000456  rAliasType = 'alis';                { Aliases are stored as resources of this type }
000457
000458  { define alias resolution action rules mask }
000459  kARMMountVol = $00000001;           { mount the volume automatically }
000460  kARMNoUI = $00000002;               { no user interface allowed during resolution }
000461  kARMMultVols = $00000008;           { search on multiple volumes }
000462  kARMSearch = $00000100;             { search quickly }
000463  kARMSearchMore = $00000200;         { search further }
000464  kARMSearchRelFirst = $00000400;     { search target on a relative path first }
000465
000466  { define alias record information types }
```

```
000467  asiZoneName = -3;                  { get zone name }
000468  asiServerName = -2;                { get server name }
000469  asiVolumeName = -1;                { get volume name }
000470  asiAliasName = 0;                  { get aliased file/folder/volume name }
000471  asiParentName = 1;                 { get parent folder name }
000472
000473
000474  TYPE
000475  { define the alias record that will be the blackbox for the caller }
000476  AliasPtr = ^AliasRecord;
000477  AliasHandle = ^AliasPtr;
000478  AliasRecord = RECORD
000479      userType: OSType;              { appl stored type like creator type }
000480      aliasSize: INTEGER;            { alias record size in bytes, for appl usage }
000481      END;
000482
000483
000484  AliasInfoType = INTEGER;           { alias record information type }
000485  AliasFilterProcPtr = ProcPtr;
000486
000487
000488  {  create a new alias between fromFile-target and return alias record handle  }
000489  FUNCTION NewAlias(fromFile: FSSpecPtr;
000490                    target: FSSpec;
000491                    VAR alias: AliasHandle): OSErr;
000492      INLINE $7002,$A823;
000493
000494  { create a minimal new alias for a target and return alias record handle }
000495  FUNCTION NewAliasMinimal(target: FSSpec;
000496                           VAR alias: AliasHandle): OSErr;
000497      INLINE $7008,$A823;
000498
000499  { create a minimal new alias from a target fullpath (optional zone and server name) and return alias record handle  }
000500  FUNCTION NewAliasMinimalFromFullPath(fullPathLength: INTEGER;
000501                                       fullPath: Ptr;
000502                                       zoneName: Str32;
000503                                       serverName: Str31;
000504                                       VAR alias: AliasHandle): OSErr;
000505      INLINE $7009,$A823;
000506
000507  { given an alias handle and fromFile, resolve the alias, update the alias record and return aliased filename and wasChanged flag. }
000508  FUNCTION ResolveAlias(fromFile: FSSpecPtr;
000509                        alias: AliasHandle;
000510                        VAR target: FSSpec;
000511                        VAR wasChanged: BOOLEAN): OSErr;
000512      INLINE $7003,$A823;
000513
000514  { given an alias handle and an index specifying requested alias information type, return the information from alias record as a string. }
000515  FUNCTION GetAliasInfo(alias: AliasHandle;
000516                        index: AliasInfoType;
000517                        VAR theString: Str63): OSErr;
000518      INLINE $7007,$A823;
000519
000520  {  given a file spec, return target file spec if input file spec is an alias.
000521      It resolves the entire alias chain or one step of the chain.  It returns
000522      info about whether the target is a folder or file; and whether the input
```

```
000523      file spec was an alias or not.  }
000524  FUNCTION ResolveAliasFile(VAR theSpec: FSSpec;
000525                            resolveAliasChains: BOOLEAN;
000526                            VAR targetIsFolder: BOOLEAN;
000527                            VAR wasAliased: BOOLEAN): OSErr;
000528      INLINE $700C,$A823;
000529
000530  {   Low Level Routines
000531  given an alias handle and fromFile, match the alias and return aliased filename(s) and needsUpdate flag }
000532  FUNCTION MatchAlias(fromFile: FSSpecPtr;
000533                      rulesMask: LONGINT;
000534                      alias: AliasHandle;
000535                      VAR aliasCount: INTEGER;
000536                      aliasList: FSSpecArrayPtr;
000537                      VAR needsUpdate: BOOLEAN;
000538                      aliasFilter: AliasFilterProcPtr;
000539                      yourDataPtr: UNIV Ptr): OSErr;
000540      INLINE $7005,$A823;
000541
000542  { given a fromFile-target pair and an alias handle, update the lias record pointed to by alias handle to represent target as the new alias. }
000543  FUNCTION UpdateAlias(fromFile: FSSpecPtr;
000544                       target: FSSpec;
000545                       alias: AliasHandle;
000546                       VAR wasChanged: BOOLEAN): OSErr;
000547      INLINE $7006,$A823;
000548
000549
000550  {$ENDC}    { UsingAliases }
000551
000552  {$IFC NOT UsingIncludes}
000553      END.
000554  {$ENDC}
000555
000556
000557  ### END OF FILE Aliases.p
000558
```

```
000559
000560  ############################################################################
000561  ### FILE: AppleEvents.p
000562  ############################################################################
000563
000564
000565  {
000566  Created: Monday, September 16, 1991 at 2:41 PM
000567   AppleEvents.p
000568   Pascal Interface to the Macintosh Libraries
000569
000570    Copyright Apple Computer, Inc. 1989-1991
000571    All rights reserved
000572
000573  }
000574
000575
000576  {$IFC UNDEFINED UsingIncludes}
000577  {$SETC UsingIncludes := 0}
000578  {$ENDC}
000579
000580  {$IFC NOT UsingIncludes}
000581   UNIT AppleEvents;
000582   INTERFACE
000583  {$ENDC}
000584
000585  {$IFC UNDEFINED UsingAppleEvents}
000586  {$SETC UsingAppleEvents := 1}
000587
000588  {$I+}
000589  {$SETC AppleEventsIncludes := UsingIncludes}
000590  {$SETC UsingIncludes := 1}
000591  {$IFC UNDEFINED UsingTypes}
000592  {$I $$Shell(PInterfaces)Types.p}
000593  {$ENDC}
000594  {$IFC UNDEFINED UsingMemory}
000595  {$I $$Shell(PInterfaces)Memory.p}
000596  {$ENDC}
000597  {$IFC UNDEFINED UsingOSUtils}
000598  {$I $$Shell(PInterfaces)OSUtils.p}
000599  {$ENDC}
000600  {$IFC UNDEFINED UsingEvents}
000601  {$I $$Shell(PInterfaces)Events.p}
000602  {$ENDC}
000603  {$IFC UNDEFINED UsingEPPC}
000604  {$I $$Shell(PInterfaces)EPPC.p}
000605  {$ENDC}
000606  {$IFC UNDEFINED UsingNotification}
000607  {$I $$Shell(PInterfaces)Notification.p}
000608  {$ENDC}
000609  {$SETC UsingIncludes := AppleEventsIncludes}
000610
000611  CONST
000612  typeBoolean = 'bool';
000613  typeChar = 'TEXT';
000614  typeSMInt = 'shor';
```

```
000615  typeInteger = 'long';
000616  typeSMFloat = 'sing';
000617  typeFloat = 'doub';
000618  typeLongInteger = 'long';
000619  typeShortInteger = 'shor';
000620  typeLongFloat = 'doub';
000621  typeShortFloat = 'sing';
000622  typeExtended = 'exte';
000623  typeComp = 'comp';
000624  typeMagnitude = 'magn';
000625  typeAEList = 'list';
000626  typeAERecord = 'reco';
000627  typeTrue = 'true';
000628  typeFalse = 'fals';
000629  typeAlias = 'alis';
000630  typeEnumerated = 'enum';
000631  typeType = 'type';
000632  typeAppParameters = 'appa';
000633  typeProperty = 'prop';
000634  typeFSS = 'fss ';
000635  typeKeyword = 'keyw';
000636  typeSectionH = 'sect';
000637  typeWildCard = '****';
000638
000639  typeApplSignature = 'sign';
000640  typeSessionID = 'ssid';
000641  typeTargetID = 'targ';
000642  typeProcessSerialNumber = 'psn ';
000643  typeNull = 'null';                  {the type of null/nonexistent data}
000644
000645  kCoreEventClass = 'aevt';
000646
000647  kAEOpenApplication = 'oapp';
000648  kAEOpenDocuments = 'odoc';
000649  kAEPrintDocuments = 'pdoc';
000650  kAEQuitApplication = 'quit';
000651
000652  kAECreatorType = 'crea';
000653  kAEQuitAll = 'quia';
000654  kAEShutDown = 'shut';
000655  kAERestart = 'rest';
000656  kAEApplicationDied = 'obit';
000657  keyProcessSerialNumber = 'psn ';
000658
000659  keyErrorNumber = 'errn';
000660  keyErrorString = 'errs';
000661
000662  kAEAnswer = 'ansr';
000663
000664  keyDirectObject = '----';
000665
000666
000667  { keyword used in install special handler }
000668  keyPreDispatch = 'phac';            { PreHandler Accessor Call }
000669  keySelectProc = 'selh';             { More selector Call }
000670
```

```
000671  { keywords used in attributes }
000672  keyTransactionIDAttr = 'tran';
000673  keyReturnIDAttr = 'rtid';
000674  keyEventClassAttr = 'evcl';
000675  keyEventIDAttr = 'evid';
000676  keyAddressAttr = 'addr';
000677  keyOptionalKeywordAttr = 'optk';
000678  keyTimeoutAttr = 'timo';
000679  keyInteractLevelAttr = 'inte';      {this attribute is read only will be set in AESend}
000680  keyEventSourceAttr = 'esrc';        { this attribute is read only }
000681  keyMissedKeywordAttr = 'miss';      { this attribute is read only }
000682
000683  { constants for use in AESendMode }
000684  kAENoReply = $00000001;             { Sender doesn't want a reply to event }
000685  kAEQueueReply = $00000002;          { Sender wants a reply but won't wait }
000686  kAEWaitReply = $00000003;           { Sender wants a reply and will be waiting }
000687
000688  kAENeverInteract = $00000010;       { Server should not interact with user }
000689  kAECanInteract = $00000020;         { Server may try to interact with user }
000690  kAEAlwaysInteract = $00000030;      { Server should always interact with user where appropriate }
000691
000692  kAECanSwitchLayer = $00000040;      { Interaction may switch layer }
000693
000694  kAEDontReconnect = $00000080;       { don't reconnect if there is a sessClosedErr from PPCToolbox }
000695
000696  kAEWantReceipt = nReturnReceipt;    { Send wants a receipt of message }
000697
000698  { constants to be used in AESendPriority }
000699  kAENormalPriority = $00000000;      { Post message at the end of event queue }
000700  kAEHighPriority = nAttnMsg;         { Post message at the front of the event queue }
000701
000702  { special constants in generating events }
000703  kAnyTransactionID = 0;              { no transaction is in use }
000704  kAutoGenerateReturnID = -1;         { AECreateAppleEvent will generate a session-unique ID }
000705
000706  { constant for use AESend }
000707  kAEDefaultTimeout = -1;             { timeout value determined by AEM }
000708  kNoTimeOut = -2;                    { wait until reply comes back, however long it takes }
000709
000710  { dispatch parameter to AEResumeTheCurrentEvent takes a pointer to a dispatch
000711  table, or one of these two constants }
000712  kAENoDispatch = 0;
000713  kAEUseStandardDispatch = -1;
000714
000715  { Error messages in response to reading and writing event contents }
000716  errAECoercionFail = -1700;
000717  errAEDescNotFound = -1701;
000718  errAECorruptData = -1702;
000719  errAEWrongDataType = -1703;
000720  errAENotAEDesc = -1704;
000721  errAEBadListItem = -1705;           { Specified list item does not exist }
000722  errAENewerVersion = -1706;          { Need newer version of AppleEvent Manager }
000723  errAENotAppleEvent = -1707;         { The event is not in AppleEvent format }
000724
000725  { Error messages in response to sending/receiving a message }
000726  errAEEventNotHandled = -1708;       { The AppleEvent was not handled by any handler }
```

```
000727  errAEReplyNotValid = -1709;         { AEResetTimer was passed an invalid reply parameter }
000728  errAEUnknownSendMode = -1710;       { Mode wasn't NoReply, WaitReply, or QueueReply;  or Interaction level is unknown }
000729  errAEWaitCanceled = -1711;          { In AESend, User cancelled out of wait loop for  reply or receipt }
000730  errAETimeout = -1712;               { AppleEvent timed out }
000731
000732  errAENoUserInteraction = -1713;     { no user interaction allowed }
000733  errAENotASpecialFunction = -1714;   { there is no special function with this keyword }
000734  errAEParamMissed = -1715;           { a required parameter was not accessed }
000735
000736  errAEUnknownAddressType = -1716;    { The target address type is not known }
000737  errAEHandlerNotFound = -1717;       { No handler in the dispatch tables fits the parameters to
000738               AEGetEventHandler or AEGetCoercionHandler }
000739
000740  errAEReplyNotArrived = -1718;       { the contents of the reply you are accessing have not arrived yet }
000741  errAEIllegalIndex = -1719;          { Index is out of range in a put operation }
000742
000743  TYPE
000744  AEKeyword    = PACKED ARRAY [1..4] OF CHAR;
000745  AEEventClass  = PACKED ARRAY [1..4] OF CHAR;
000746  AEEventID    = PACKED ARRAY [1..4] OF CHAR;
000747  DescType = ResType;
000748
000749  { tagged data, the standard AppleEvent data type }
000750  AEDesc = RECORD
000751   descriptorType: DescType;
000752   dataHandle: Handle;
000753   END;
000754
000755
000756  AEAddressDesc = AEDesc;             { an AEDesc which contains addressing data }
000757  AEDescList = AEDesc;                { a list of AEDesc is a special kind of AEDesc }
000758  AERecord = AEDescList;              { AERecord is a list of keyworded AEDesc }
000759  AppleEvent = AERecord;              { an AERecord that contains an AppleEvent }
000760  AESendMode = LONGINT;               { Type of parameter to AESend }
000761  AESendPriority = INTEGER;           { Type of priority param of AESend }
000762
000763  { type of param to AEGetInteractionAllowed and AESetInteractionAllowed }
000764  AEInteractAllowed = (kAEInteractWithSelf,kAEInteractWithLocal,kAEInteractWithAll);
000765
000766  { Return param to AEGetTheCurrentEvent, and kAEEventSource attribute }
000767  AEEventSource = (kAEUnknownSource,kAEDirectCall,kAESameProcess,kAELocalProcess,
000768   kAERemoteProcess);
000769
000770
000771  { types for AppleEvent Array support
000772
000773   Basic data type of attibutes & parameters}
000774  AEKeyDesc = RECORD
000775   descKey: AEKeyword;
000776   descContent: AEDesc;
000777   END;
000778
000779
000780  AEArrayType = (kAEDataArray,kAEPackedArray,kAEHandleArray,kAEDescArray,
000781   kAEKeyDescArray);
000782
```

```
000783
000784  { Array routines support these different types of elements}
000785  AEArrayData = RECORD
000786   case AEArrayType OF
000787    kAEDataArray:
000788     (AEDataArray:   Array[0..0] OF Integer);
000789    kAEPackedArray:
000790     (AEPackedArray:  Packed Array[0..0] OF Char);
000791    kAEHandleArray:
000792     (AEHandleArray:  Array[0..0] OF Handle);
000793    kAEDescArray:
000794     (AEDescArray:   Array[0..0] OF AEDesc);
000795    kAEKeyDescArray:
000796     (AEKeyDescArray:  Array[0..0] OF AEKeyDesc);
000797   END;
000798
000799  AEArrayDataPointer = ^AEArrayData;
000800
000801
000802
000803  EventHandlerProcPtr = ProcPtr;
000804  IdleProcPtr = ProcPtr;
000805  EventFilterProcPtr = ProcPtr;
000806
000807
000808  {   ****************************************************************
000809  The following calls apply to any AEDesc. Every result descriptor is created for you,
000810  so you will be responsible for memory management of the descriptors so created.
000811  Purgeable descriptor data is not supported: the AEM does not call LoadResource.   }
000812  FUNCTION AECreateDesc(typeCode: DescType;
000813                        dataPtr: Ptr;
000814                        dataSize: Size;
000815                        VAR result: AEDesc): OSErr;
000816   INLINE $303C, $0825, $A816;
000817  FUNCTION AECoercePtr(typeCode: DescType;
000818                       dataPtr: Ptr;
000819                       dataSize: Size;
000820                       toType: DescType;
000821                       VAR result: AEDesc): OSErr;
000822   INLINE $303C, $0A02, $A816;
000823  FUNCTION AECoerceDesc(theAEDesc: AEDesc;
000824                        toType: DescType;
000825                        VAR result: AEDesc): OSErr;
000826   INLINE $303C, $0603, $A816;
000827  FUNCTION AEDisposeDesc(VAR theAEDesc: AEDesc): OSErr;
000828   INLINE $303C, $0204, $A816;
000829  FUNCTION AEDuplicateDesc(theAEDesc: AEDesc;
000830                           VAR result: AEDesc): OSErr;
000831   INLINE $303C, $0405, $A816;
000832
000833  { ****************************************************************
000834   The following calls apply to AEDescList.
000835    Since AEDescList is a subtype of AEDesc, the calls in the previous
000836    section can also be used for AEDescList. All list and array indices are 1-based.
000837    If the data was greater than maximumSize in the routines below, then actualSize will
000838    be greater than maximumSize, but only maximumSize bytes will actually be retrieved. }
```

```
000839  FUNCTION AECreateList(factoringPtr: Ptr;
000840                        factoredSize: Size;
000841                        isRecord: BOOLEAN;
000842                        VAR resultList: AEDescList): OSErr;
000843   INLINE $303C, $0706, $A816;
000844  FUNCTION AECountItems(theAEDescList: AEDescList;
000845                        VAR theCount: LONGINT): OSErr;
000846   INLINE $303C, $0407, $A816;
000847  FUNCTION AEPutPtr(theAEDescList: AEDescList;
000848                    index: LONGINT;
000849                    typeCode: DescType;
000850                    dataPtr: Ptr;
000851                    dataSize: Size): OSErr;
000852   INLINE $303C, $0A08, $A816;
000853  FUNCTION AEPutDesc(theAEDescList: AEDescList;
000854                     index: LONGINT;
000855                     theAEDesc: AEDesc): OSErr;
000856   INLINE $303C, $0609, $A816;
000857  FUNCTION AEGetNthPtr(theAEDescList: AEDescList;
000858                       index: LONGINT;
000859                       desiredType: DescType;
000860                       VAR theAEKeyword: AEKeyword;
000861                       VAR typeCode: DescType;
000862                       dataPtr: Ptr;
000863                       maximumSize: Size;
000864                       VAR actualSize: Size): OSErr;
000865   INLINE $303C, $100A, $A816;
000866  FUNCTION AEGetNthDesc(theAEDescList: AEDescList;
000867                        index: LONGINT;
000868                        desiredType: DescType;
000869                        VAR theAEKeyword: AEKeyword;
000870                        VAR result: AEDesc): OSErr;
000871   INLINE $303C, $0A0B, $A816;
000872  FUNCTION AESizeOfNthItem(theAEDescList: AEDescList;
000873                           index: LONGINT;
000874                           VAR typeCode: DescType;
000875                           VAR dataSize: Size): OSErr;
000876   INLINE $303C, $082A, $A816;
000877  FUNCTION AEGetArray(theAEDescList: AEDescList;
000878                      arrayType: AEArrayType;
000879                      arrayPtr: AEArrayDataPointer;
000880                      maximumSize: Size;
000881                      VAR itemType: DescType;
000882                      VAR itemSize: Size;
000883                      VAR itemCount: LONGINT): OSErr;
000884   INLINE $303C, $0D0C, $A816;
000885  FUNCTION AEPutArray(theAEDescList: AEDescList;
000886                      arrayType: AEArrayType;
000887                      arrayPtr: AEArrayDataPointer;
000888                      itemType: DescType;
000889                      itemSize: Size;
000890                      itemCount: LONGINT): OSErr;
000891   INLINE $303C, $0B0D, $A816;
000892  FUNCTION AEDeleteItem(theAEDescList: AEDescList;
000893                        index: LONGINT): OSErr;
000894   INLINE $303C, $040E, $A816;
```

```
000895
000896  { ************************************************************************
000897  The following calls apply to AERecord.
000898    Since AERecord is a subtype of AEDescList, the calls in the previous
000899    sections can also be used for AERecord
000900    an AERecord can be created by using AECreateList with isRecord set to true  }
000901  FUNCTION AEPutKeyPtr(theAERecord: AERecord;
000902                       theAEKeyword: AEKeyword;
000903                       typeCode: DescType;
000904                       dataPtr: Ptr;
000905                       dataSize: Size): OSErr;
000906   INLINE $303C, $0A0F, $A816;
000907  FUNCTION AEPutKeyDesc(theAERecord: AERecord;
000908                        theAEKeyword: AEKeyword;
000909                        theAEDesc: AEDesc): OSErr;
000910   INLINE $303C, $0610, $A816;
000911  FUNCTION AEGetKeyPtr(theAERecord: AERecord;
000912                       theAEKeyword: AEKeyword;
000913                       desiredType: DescType;
000914                       VAR typeCode: DescType;
000915                       dataPtr: Ptr;
000916                       maximumSize: Size;
000917                       VAR actualSize: Size): OSErr;
000918   INLINE $303C, $0E11, $A816;
000919  FUNCTION AEGetKeyDesc(theAERecord: AERecord;
000920                        theAEKeyword: AEKeyword;
000921                        desiredType: DescType;
000922                        VAR result: AEDesc): OSErr;
000923   INLINE $303C, $0812, $A816;
000924  FUNCTION AESizeOfKeyDesc(theAERecord: AERecord;
000925                           theAEKeyword: AEKeyword;
000926                           VAR typeCode: DescType;
000927                           VAR dataSize: Size): OSErr;
000928   INLINE $303C, $0829, $A816;
000929  FUNCTION AEDeleteKeyDesc(theAERecord: AERecord;
000930                           theAEKeyword: AEKeyword): OSErr;
000931   INLINE $303C, $0413, $A816;
000932
000933  {
000934   ************************************************************************
000935   The following calls are used to pack and unpack parameters from records of
000936   type AppleEvent. Since AppleEvent is a subtype of AERecord, the calls in the previous
000937   sections can also be used for variables of type AppleEvent. The next six calls
000938   are in fact identical to the six calls for AERecord.
000939   }
000940  FUNCTION AEPutParamPtr(theAppleEvent: AppleEvent;
000941                         theAEKeyword: AEKeyword;
000942                         typeCode: DescType;
000943                         dataPtr: Ptr;
000944                         dataSize: Size): OSErr;
000945   INLINE $303C,$0A0F,$A816;
000946  FUNCTION AEPutParamDesc(theAppleEvent: AppleEvent;
000947                          theAEKeyword: AEKeyword;
000948                          theAEDesc: AEDesc): OSErr;
000949   INLINE $303C,$0610,$A816;
000950  FUNCTION AEGetParamPtr(theAppleEvent: AppleEvent;
```

```
000951                          theAEKeyword: AEKeyword;
000952                          desiredType: DescType;
000953                          VAR typeCode: DescType;
000954                          dataPtr: Ptr;
000955                          maximumSize: Size;
000956                          VAR actualSize: Size): OSErr;
000957   INLINE $303C,$0E11,$A816;
000958  FUNCTION AEGetParamDesc(theAppleEvent: AppleEvent;
000959                           theAEKeyword: AEKeyword;
000960                           desiredType: DescType;
000961                           VAR result: AEDesc): OSErr;
000962   INLINE $303C,$0812,$A816;
000963  FUNCTION AESizeOfParam(theAppleEvent: AppleEvent;
000964                          theAEKeyword: AEKeyword;
000965                          VAR typeCode: DescType;
000966                          VAR dataSize: Size): OSErr;
000967   INLINE $303C,$0829,$A816;
000968  FUNCTION AEDeleteParam(theAppleEvent: AppleEvent;
000969                          theAEKeyword: AEKeyword): OSErr;
000970   INLINE $303C,$0413,$A816;
000971
000972  { ********************************************************************
000973  The following calls also apply to type AppleEvent.  Message attributes are far more restricted, and
000974  can only be accessed through the following 5 calls. The various list and record routines cannot be used
000975  to access the attributes of an event.  }
000976  FUNCTION AEGetAttributePtr(theAppleEvent: AppleEvent;
000977                             theAEKeyword: AEKeyword;
000978                             desiredType: DescType;
000979                             VAR typeCode: DescType;
000980                             dataPtr: Ptr;
000981                             maximumSize: Size;
000982                             VAR actualSize: Size): OSErr;
000983   INLINE $303C,$0E15,$A816;
000984  FUNCTION AEGetAttributeDesc(theAppleEvent: AppleEvent;
000985                              theAEKeyword: AEKeyword;
000986                              desiredType: DescType;
000987                              VAR result: AEDesc): OSErr;
000988   INLINE $303C,$0826,$A816;
000989  FUNCTION AESizeOfAttribute(theAppleEvent: AppleEvent;
000990                             theAEKeyword: AEKeyword;
000991                             VAR typeCode: DescType;
000992                             VAR dataSize: Size): OSErr;
000993   INLINE $303C,$0828,$A816;
000994  FUNCTION AEPutAttributePtr(theAppleEvent: AppleEvent;
000995                             theAEKeyword: AEKeyword;
000996                             typeCode: DescType;
000997                             dataPtr: Ptr;
000998                             dataSize: Size): OSErr;
000999   INLINE $303C,$0A16,$A816;
001000  FUNCTION AEPutAttributeDesc(theAppleEvent: AppleEvent;
001001                              theAEKeyword: AEKeyword;
001002                              theAEDesc: AEDesc): OSErr;
001003   INLINE $303C,$0627,$A816;
001004
001005  { ********************************************************************
001006   The next four calls are basic routines used to create, send, and process AppleEvents.  }
```

```
001007  FUNCTION AECreateAppleEvent(theAEEventClass: AEEventClass;
001008                              theAEEventID: AEEventID;
001009                              target: AEAddressDesc;
001010                              returnID: INTEGER;
001011                              transactionID: LONGINT;
001012                              VAR result: AppleEvent): OSErr;
001013   INLINE $303C,$0B14,$A816;
001014  FUNCTION AESend(theAppleEvent: AppleEvent;
001015                  VAR reply: AppleEvent;
001016                  sendMode: AESendMode;
001017                  sendPriority: AESendPriority;
001018                  timeOutInTicks: LONGINT;
001019                  idleProc: IdleProcPtr;
001020                  filterProc: EventFilterProcPtr): OSErr;
001021   INLINE $303C,$0D17,$A816;
001022  FUNCTION AEProcessAppleEvent(theEventRecord: EventRecord): OSErr;
001023   INLINE $303C,$021B,$A816;
001024
001025  { During event processing, an event handler may realize that it is likely
001026  to exceed the client's timeout limit. Passing the reply to this
001027  routine causes a wait event to be generated to ask the client for more time.  }
001028  FUNCTION AEResetTimer(reply: AppleEvent): OSErr;
001029   INLINE $303C,$0219,$A816;
001030
001031  {   ************************************************************************
001032  The following four calls are available for applications which need more sophisticated control
001033  over when and how events are processed. Applications which implement multi-session servers or
001034  which implement their own internal event queueing will probably be the major clients of these
001035  routines.
001036
001037   Can be called from within a handler to prevent the AEM from disposing of
001038  the AppleEvent when the handler returns. Can be used to asynchronously process the
001039  event (as in MacApp). }
001040  FUNCTION AESuspendTheCurrentEvent(theAppleEvent: AppleEvent): OSErr;
001041   INLINE $303C,$022B,$A816;
001042
001043  {
001044    Tells the AppleEvent manager that processing is either about to resume or has
001045   been completed on a previously suspended event. The procPtr passed in as the
001046   dispatcher parameter will be called to attempt to redispatch the event. Several
001047   constants for the dispatcher parameter allow special behavior. They are:
001048   - kAEUseStandardDispatch means redispatch as if the event was just received, using the
001049    standard AppleEvent Dispatcher.
001050   - kAENoDispatch means ignore the parameter.
001051    Use this in the case where no redispatch is needed, and the event has been handled.
001052   - non nil means call the routine which dispatcher points to.
001053   }
001054  FUNCTION AEResumeTheCurrentEvent(theAppleEvent: AppleEvent;
001055                                   reply: AppleEvent;
001056                                   dispatcher: EventHandlerProcPtr;
001057                                   handlerRefcon: LONGINT): OSErr;
001058   INLINE $303C,$0818,$A816;
001059
001060  { Allows application to examine the currently executing event }
001061  FUNCTION AEGetTheCurrentEvent(VAR theAppleEvent: AppleEvent): OSErr;
001062   INLINE $303C,$021A,$A816;
```

```
001063
001064  { Set the current event to the parameter }
001065  FUNCTION AESetTheCurrentEvent(theAppleEvent: AppleEvent): OSErr;
001066   INLINE $303C,$022C,$A816;
001067
001068  {
001069   ********************************************************************
001070    The following three calls are used to allow applications to behave courteously
001071   when a user interaction such as a dialog box is needed.
001072   }
001073  FUNCTION AEGetInteractionAllowed(VAR level: AEInteractAllowed): OSErr;
001074   INLINE $303C,$021D,$A816;
001075  FUNCTION AESetInteractionAllowed(level: AEInteractAllowed): OSErr;
001076   INLINE $303C,$011E,$A816;
001077  FUNCTION AEInteractWithUser(timeOutInTicks: LONGINT;
001078                              nmReqPtr: NMRecPtr;
001079                              idleProc: IdleProcPtr): OSErr;
001080   INLINE $303C,$061C,$A816;
001081
001082  {  ********************************************************************
001083   These calls are used to set up and modify the event dispatch table  }
001084
001085  { Add an AppleEvent Handler  }
001086  FUNCTION AEInstallEventHandler(theAEEventClass: AEEventClass;
001087                                 theAEEventID: AEEventID;
001088                                 handler: EventHandlerProcPtr;
001089                                 handlerRefcon: LONGINT;
001090                                 isSysHandler: BOOLEAN): OSErr;
001091   INLINE $303C,$091F,$A816;
001092
001093  {  Remove an AppleEvent Handler  }
001094  FUNCTION AERemoveEventHandler(theAEEventClass: AEEventClass;
001095                                theAEEventID: AEEventID;
001096                                handler: EventHandlerProcPtr;
001097                                isSysHandler: BOOLEAN): OSErr;
001098   INLINE $303C,$0720,$A816;
001099
001100  {  Get the corresponding AppleEvent Handler  }
001101  FUNCTION AEGetEventHandler(theAEEventClass: AEEventClass;
001102                             theAEEventID: AEEventID;
001103                             VAR handler: EventHandlerProcPtr;
001104                             VAR handlerRefcon: LONGINT;
001105                             isSysHandler: BOOLEAN): OSErr;
001106   INLINE $303C,$0921,$A816;
001107
001108  {  ********************************************************************
001109   These calls are used to set up and modify the coercion dispatch table   }
001110  FUNCTION AEInstallCoercionHandler(fromType: DescType;
001111                                    toType: DescType;
001112                                    handler: ProcPtr;
001113                                    handlerRefcon: LONGINT;
001114                                    fromTypeIsDesc: BOOLEAN;
001115                                    isSysHandler: BOOLEAN): OSErr;
001116   INLINE $303C, $0A22, $A816;
001117
001118  {  Remove a Coercion Handler  }
```

```
001119  FUNCTION AERemoveCoercionHandler(fromType: DescType;
001120                                   toType: DescType;
001121                                   handler: ProcPtr;
001122                                   isSysHandler: BOOLEAN): OSErr;
001123   INLINE $303C, $0723, $A816;
001124
001125  {  Get the corresponding Coercion Handler  }
001126  FUNCTION AEGetCoercionHandler(fromType: DescType;
001127                                toType: DescType;
001128                                VAR handler: ProcPtr;
001129                                VAR handlerRefcon: LONGINT;
001130                                VAR fromTypeIsDesc: BOOLEAN;
001131                                isSysHandler: BOOLEAN): OSErr;
001132   INLINE $303C, $0B24, $A816;
001133
001134  {
001135     ************************************************************************
001136   These calls are used to set up and modify special hooks into the AppleEvent Manager.
001137
001138   Install the special handler named by the Keyword   }
001139  FUNCTION AEInstallSpecialHandler(functionClass: AEKeyword;
001140                                   handler: ProcPtr;
001141                                   isSysHandler: BOOLEAN): OSErr;
001142   INLINE $303C, $0500, $A816;
001143
001144  {  Remove the special handler named by the Keyword  }
001145  FUNCTION AERemoveSpecialHandler(functionClass: AEKeyword;
001146                                  handler: ProcPtr;
001147                                  isSysHandler: BOOLEAN): OSErr;
001148   INLINE $303C, $0501, $A816;
001149
001150  {  Get the special handler named by the Keyword  }
001151  FUNCTION AEGetSpecialHandler(functionClass: AEKeyword;
001152                               VAR handler: ProcPtr;
001153                               isSysHandler: BOOLEAN): OSErr;
001154   INLINE $303C, $052D, $A816;
001155
001156
001157  {$ENDC} { UsingAppleEvents }
001158
001159  {$IFC NOT UsingIncludes}
001160   END.
001161  {$ENDC}
001162
001163
001164  ### END OF FILE AppleEvents.p
001165
```

```
001166
001167  ############################################################################
001168  ### FILE: AppleTalk.p
001169  ############################################################################
001170
001171
001172  {
001173  Created: Sunday, September 15, 1991 at 9:47 PM
001174   AppleTalk.p
001175   Pascal Interface to the Macintosh Libraries
001176
001177    Copyright Apple Computer, Inc. 1985-1991
001178    All rights reserved
001179  }
001180
001181
001182  {$IFC UNDEFINED UsingIncludes}
001183  {$SETC UsingIncludes := 0}
001184  {$ENDC}
001185
001186  {$IFC NOT UsingIncludes}
001187   UNIT AppleTalk;
001188   INTERFACE
001189  {$ENDC}
001190
001191  {$IFC UNDEFINED UsingAppleTalk}
001192  {$SETC UsingAppleTalk := 1}
001193
001194  {$I+}
001195  {$SETC AppleTalkIncludes := UsingIncludes}
001196  {$SETC UsingIncludes := 1}
001197  {$IFC UNDEFINED UsingTypes}
001198  {$I $$Shell(PInterfaces)Types.p}
001199  {$ENDC}
001200  {$IFC UNDEFINED UsingOSUtils}
001201  {$I $$Shell(PInterfaces)OSUtils.p}
001202  {$ENDC}
001203  {$SETC UsingIncludes := AppleTalkIncludes}
001204
001205  CONST
001206
001207  { Driver unit and reference numbers (ADSP is dynamic) }
001208
001209  mppUnitNum = 9;                               { MPP unit number }
001210  atpUnitNum = 10;                              { ATP unit number }
001211  xppUnitNum = 40;                              { XPP unit number }
001212  mppRefNum = -10;                              { MPP reference number }
001213  atpRefNum = -11;                              { ATP reference number }
001214  xppRefNum = -41;                              { XPP reference number }
001215
001216  { .MPP csCodes }
001217
001218  lookupReply = 242;                            { This command queued to ourself }
001219  writeLAP = 243;                               { Write out LAP packet }
001220  detachPH = 244;                               { Detach LAP protocol handler }
001221  attachPH = 245;                               { Attach LAP protocol handler }
```

```
001222  writeDDP = 246;                         { Write out DDP packet }
001223  closeSkt = 247;                         { Close DDP socket }
001224  openSkt = 248;                          { Open DDP socket }
001225  loadNBP = 249;                          { Load NBP command-executing code }
001226  lastResident = 249;                     { Last resident command }
001227  confirmName = 250;                      { Confirm name }
001228  lookupName = 251;                       { Look up name on internet }
001229  removeName = 252;                       { Remove name from Names Table }
001230  registerName = 253;                     { Register name in Names Table }
001231  killNBP = 254;                          { Kill outstanding NBP request }
001232  unloadNBP = 255;                        { Unload NBP command code }
001233  setSelfSend = 256;                      { MPP: Set to allow writes to self }
001234  SetMyZone = 257;                        { Set my zone name }
001235  GetATalkInfo = 258;                     { get AppleTalk information }
001236  ATalkClosePrep = 259;                   { AppleTalk close query }
001237
001238  { .ATP csCodes }
001239
001240  nSendRequest = 248;                     { NSendRequest code }
001241  relRspCB = 249;                         { Release RspCB }
001242  closeATPSkt = 250;                      { Close ATP socket }
001243  addResponse = 251;                      { Add response code | Require open skt }
001244  sendResponse = 252;                     { Send response code }
001245  getRequest = 253;                       { Get request code }
001246  openATPSkt = 254;                       { Open ATP socket }
001247  sendRequest = 255;                      { Send request code }
001248  relTCB = 256;                           { Release TCB }
001249  killGetReq = 257;                       { Kill GetRequest }
001250  killSendReq = 258;                      { Kill SendRequest }
001251  killAllGetReq = 259;                    { Kill all getRequests for a skt }
001252
001253  { .XPP csCodes }
001254
001255  openSess = 255;                         { Open session }
001256  closeSess = 254;                        { Close session }
001257  userCommand = 253;                      { User command }
001258  userWrite = 252;                        { User write }
001259  getStatus = 251;                        { Get status }
001260  afpCall = 250;                          { AFP command (buffer has command code) }
001261  getParms = 249;                         { Get parameters }
001262  abortOS = 248;                          { Abort open session request }
001263  closeAll = 247;                         { Close all open sessions }
001264  xCall = 246;                            { .XPP extended calls }
001265
001266  { Transition Queue transition types }
001267  ATTransOpen = 0;                        {AppleTalk has opened}
001268  ATTransClose = 2;                       {AppleTalk is about to close}
001269  ATTransClosePrep = 3;                   {Is it OK to close AppleTalk ?}
001270  ATTransCancelClose = 4;                 {Cancel the ClosePrep transition}
001271
001272  afpByteRangeLock = 1;                   {AFPCall command codes}
001273  afpVolClose = 2;                        {AFPCall command codes}
001274  afpDirClose = 3;                        {AFPCall command codes}
001275  afpForkClose = 4;                       {AFPCall command codes}
001276  afpCopyFile = 5;                        {AFPCall command codes}
001277  afpDirCreate = 6;                       {AFPCall command codes}
```

```
001278  afpFileCreate = 7;                    {AFPCall command codes}
001279  afpDelete = 8;                        {AFPCall command codes}
001280  afpEnumerate = 9;                     {AFPCall command codes}
001281  afpFlush = 10;                        {AFPCall command codes}
001282  afpForkFlush = 11;                    {AFPCall command codes}
001283  afpGetDirParms = 12;                  {AFPCall command codes}
001284  afpGetFileParms = 13;                 {AFPCall command codes}
001285  afpGetForkParms = 14;                 {AFPCall command codes}
001286  afpGetSInfo = 15;                     {AFPCall command codes}
001287  afpGetSParms = 16;                    {AFPCall command codes}
001288  afpGetVolParms = 17;                  {AFPCall command codes}
001289  afpLogin = 18;                        {AFPCall command codes}
001290  afpContLogin = 19;                    {AFPCall command codes}
001291  afpLogout = 20;                       {AFPCall command codes}
001292  afpMapID = 21;                        {AFPCall command codes}
001293  afpMapName = 22;                      {AFPCall command codes}
001294  afpMove = 23;                         {AFPCall command codes}
001295  afpOpenVol = 24;                      {AFPCall command codes}
001296  afpOpenDir = 25;                      {AFPCall command codes}
001297  afpOpenFork = 26;                     {AFPCall command codes}
001298  afpRead = 27;                         {AFPCall command codes}
001299  afpRename = 28;                       {AFPCall command codes}
001300  afpSetDirParms = 29;                  {AFPCall command codes}
001301  afpSetFileParms = 30;                 {AFPCall command codes}
001302  afpSetForkParms = 31;                 {AFPCall command codes}
001303  afpSetVolParms = 32;                  {AFPCall command codes}
001304  afpWrite = 33;                        {AFPCall command codes}
001305  afpGetFlDrParms = 34;                 {AFPCall command codes}
001306  afpSetFlDrParms = 35;                 {AFPCall command codes}
001307  afpDTOpen = 48;                       {AFPCall command codes}
001308  afpDTClose = 49;                      {AFPCall command codes}
001309  afpGetIcon = 51;                      {AFPCall command codes}
001310  afpGtIcnInfo = 52;                    {AFPCall command codes}
001311  afpAddAPPL = 53;                      {AFPCall command codes}
001312  afpRmvAPPL = 54;                      {AFPCall command codes}
001313  afpGetAPPL = 55;                      {AFPCall command codes}
001314  afpAddCmt = 56;                       {AFPCall command codes}
001315  afpRmvCmt = 57;                       {AFPCall command codes}
001316  afpGetCmt = 58;                       {AFPCall command codes}
001317  afpAddIcon = 192;                     {Special code for ASP Write commands}
001318
001319  xppLoadedBit = 5;                     { XPP bit in PortBUse }
001320  scbMemSize = 192;                     { Size of memory for SCB }
001321  xppFlagClr = 0;                       { Cs for AFPCommandBlock }
001322  xppFlagSet = 128;                     { StartEndFlag & NewLineFlag fields. }
001323
001324  lapSize = 20;
001325  ddpSize = 26;
001326  nbpSize = 26;
001327  atpSize = 56;
001328
001329  atpXOvalue = 32;                      {ATP exactly-once bit }
001330  atpEOMvalue = 16;                     {ATP End-Of-Message bit }
001331  atpSTSvalue = 8;                      {ATP Send-Transmission-Status bit }
001332  atpTIDValidvalue = 2;                 {ATP trans. ID valid bit }
001333  atpSendChkvalue = 1;                  {ATP send checksum bit }
```

```
001334
001335  zipGetLocalZones = 5;
001336  zipGetZoneList = 6;
001337  zipGetMyZone = 7;
001338
001339  LAPMgrPtr = $B18;                               {Entry point for LAP Manager}
001340  LAPMgrCall = 2;                                 {Offset to LAP routines}
001341  LAddAEQ = 23;                                   {LAPAddATQ routine selector}
001342  LRmvAEQ = 24;                                   {LAPRmvATQ routine selector}
001343
001344  TYPE
001345  ABCallType = (tLAPRead,tLAPWrite,tDDPRead,tDDPWrite,tNBPLookup,tNBPConfirm,
001346   tNBPRegister,tATPSndRequest,tATPGetRequest,tATPSdRsp,tATPAddRsp,tATPRequest,
001347   tATPResponse);
001348
001349  ABProtoType = (lapProto,ddpProto,nbpProto,atpProto);
001350
001351
001352  ABByte = 1..127;
001353
001354
001355  LAPAdrBlock = PACKED RECORD
001356   dstNodeID: Byte;
001357   srcNodeID: Byte;
001358   lapProtType: ABByte;
001359   END;
001360
001361  ATQEntryPtr = ^ATQEntry;
001362  ATQEntry = RECORD
001363   qLink: ATQEntryPtr;                            {next queue entry}
001364   qType: INTEGER;                                {queue type}
001365   CallAddr: ProcPtr;                             {pointer to your routine}
001366   END;
001367
001368  AddrBlock = PACKED RECORD
001369   aNet: INTEGER;
001370   aNode: Byte;
001371   aSocket: Byte;
001372   END;
001373
001374  { Real definition of EntityName is 3 PACKED strings of any length (32 is just an example). No
001375  offests for Asm since each String address must be calculated by adding length byte to last string ptr.
001376  In Pascal, String(32) will be 34 bytes long since fields never start on an odd byte unless they are
001377  only a byte long. So this will generate correct looking interfaces for Pascal and C, but they will not
001378  be the same, which is OK since they are not used. }
001379  EntityPtr = ^EntityName;
001380  EntityName = RECORD
001381   objStr: Str32;
001382   typeStr: Str32;
001383   zoneStr: Str32;
001384   END;
001385
001386  RetransType = PACKED RECORD
001387   retransInterval: Byte;
001388   retransCount: Byte;
001389   END;
```

```
001390
001391  BDSElement = RECORD
001392   buffSize: INTEGER;
001393   buffPtr: Ptr;
001394   dataSize: INTEGER;
001395   userBytes: LONGINT;
001396   END;
001397
001398
001399  BDSType = ARRAY [0..7] OF BDSElement;
001400  BDSPtr = ^BDSType;
001401  BitMapType = PACKED ARRAY [0..7] OF BOOLEAN;
001402
001403  ABRecPtr = ^ABusRecord;
001404  ABRecHandle = ^ABRecPtr;
001405  ABusRecord = RECORD
001406   abOpcode: ABCallType;
001407   abResult: INTEGER;
001408   abUserReference: LONGINT;
001409   CASE ABProtoType OF
001410     lapProto:
001411    (lapAddress: LAPAdrBlock;
001412    lapReqCount: INTEGER;
001413    lapActCount: INTEGER;
001414    lapDataPtr: Ptr);
001415     ddpProto:
001416    (ddpType: Byte;
001417    ddpSocket: Byte;
001418    ddpAddress: AddrBlock;
001419    ddpReqCount: INTEGER;
001420    ddpActCount: INTEGER;
001421    ddpDataPtr: Ptr;
001422    ddpNodeID: Byte);
001423     nbpProto:
001424    (nbpEntityPtr: EntityPtr;
001425    nbpBufPtr: Ptr;
001426    nbpBufSize: INTEGER;
001427    nbpDataField: INTEGER;
001428    nbpAddress: AddrBlock;
001429    nbpRetransmitInfo: RetransType);
001430     atpProto:
001431    (atpSocket: Byte;
001432    atpAddress: AddrBlock;
001433    atpReqCount: INTEGER;
001434    atpDataPtr: Ptr;
001435    atpRspBDSPtr: BDSPtr;
001436    atpBitmap: BitMapType;
001437    atpTransID: INTEGER;
001438    atpActCount: INTEGER;
001439    atpUserData: LONGINT;
001440    atpXO: BOOLEAN;
001441    atpEOM: BOOLEAN;
001442    atpTimeOut: Byte;
001443    atpRetries: Byte;
001444    atpNumBufs: Byte;
001445    atpNumRsp: Byte;
```

```
001446    atpBDSSize: Byte;
001447    atpRspUData: LONGINT;
001448    atpRspBuf: Ptr;
001449    atpRspSize: INTEGER);
001450   END;
001451
001452  AFPCommandBlock = PACKED RECORD
001453   cmdByte: Byte;
001454   startEndFlag: Byte;
001455   forkRefNum: INTEGER;
001456   rwOffset: LONGINT;
001457   reqCount: LONGINT;
001458   newLineFlag: Byte;
001459   newLineChar: CHAR;
001460   END;
001461
001462  WDSElement = RECORD
001463   entryLength: INTEGER;
001464   entryPtr: Ptr;
001465   END;
001466
001467  NamesTableEntry = RECORD
001468   qLink: QElemPtr;
001469   nteAddress: AddrBlock;
001470   nteData: PACKED ARRAY [1..100] OF CHAR;
001471   END;
001472
001473  MPPParmType = (LAPWriteParm,AttachPHParm,DetachPHParm,OpenSktParm,CloseSktParm,
001474   WriteDDPParm,OpenATPSktParm,CloseATPSktParm,SendRequestParm,GetRequestParm,
001475   SendResponseParm,AddResponseParm,RelTCBParm,RelRspCBParm,RegisterNameParm,
001476   LookupNameParm,ConfirmNameParm,RemoveNameParm,SetSelfSendParm,NSendRequestParm,
001477   KillSendReqParm,KillGetReqParm,KillNBPParm,GetAppleTalkInfoParm,KillAllGetReqParm,
001478   ATalkClosePrepParm,CancelATalkClosePrepParm);
001479
001480  MPPPBPtr = ^MPPParamBlock;
001481  MPPParamBlock = PACKED RECORD
001482   qLink: QElemPtr;                          {next queue entry}
001483   qType: INTEGER;                           {queue type}
001484   ioTrap: INTEGER;                          {routine trap}
001485   ioCmdAddr: Ptr;                           {routine address}
001486   ioCompletion: ProcPtr;                    {completion routine}
001487   ioResult: OSErr;                          {result code}
001488   ioNamePtr: StringPtr;                     {->filename}
001489   ioVRefNum: INTEGER;                       {volume reference or drive number}
001490   ioRefNum: INTEGER;                        {driver reference number}
001491   csCode: INTEGER;                          {call command code AUTOMATICALLY set}
001492   CASE MPPParmType OF
001493     LAPWriteParm:
001494    (filler0: INTEGER;
001495    wdsPointer: Ptr);                        {->Write Data Structure}
001496     AttachPHParm,DetachPHParm:
001497    (protType: Byte;                          {ALAP Protocol Type}
001498    filler1: Byte;
001499    handler: Ptr);                           {->protocol handler routine}
001500     OpenSktParm,CloseSktParm,WriteDDPParm:
001501    (socket: Byte;                            {socket number}
```

```
001502    checksumFlag: Byte;                          {checksum flag}
001503    listener: Ptr);                              {->socket listener routine}
001504     RegisterNameParm,LookupNameParm,ConfirmNameParm,RemoveNameParm:
001505    (interval: Byte;                             {retry interval}
001506    count: Byte;                                 {retry count}
001507    entityPtr: Ptr;                              {->names table element or ->entity name}
001508    CASE MPPParmType OF
001509      RegisterNameParm:
001510     (verifyFlag: Byte;                          {set if verify needed}
001511     filler3: Byte);
001512      LookupNameParm:
001513     (retBuffPtr: Ptr;                           {->return buffer}
001514     retBuffSize: INTEGER;                       {return buffer size}
001515     maxToGet: INTEGER;                          {matches to get}
001516     numGotten: INTEGER);                        {matched gotten}
001517      ConfirmNameParm:
001518     (confirmAddr: AddrBlock;                    {->entity}
001519     newSocket: Byte;                            {socket number}
001520     filler4: Byte));
001521     SetSelfSendParm:
001522    (newSelfFlag: Byte;                          {self-send toggle flag}
001523    oldSelfFlag: Byte);                          {previous self-send state}
001524     KillNBPParm:
001525    (nKillQEl: Ptr);                             {ptr to Q element to cancel}
001526     GetAppleTalkInfoParm:
001527    (version: INTEGER;                           {requested info version}
001528    varsPtr: Ptr;                                {pointer to well known MPP vars}
001529    DCEPtr: Ptr;                                 {pointer to MPP DCE}
001530    portID: INTEGER;                             {port number [0..7]}
001531    configuration: LONGINT;                      {32-bit configuration word}
001532    selfSend: INTEGER;                           {non zero if SelfSend enabled}
001533    netLo: INTEGER;                              {low value of network range}
001534    netHi: INTEGER;                              {high value of network range}
001535    ourAddr: LONGINT;                            {our 24-bit AppleTalk address}
001536    routerAddr: LONGINT;                         {24-bit address of (last) router}
001537    numOfPHs: INTEGER;                           {max. number of protocol handlers}
001538    numOfSkts: INTEGER;                          {max. number of static sockets}
001539    numNBPEs: INTEGER;                           {max. concurrent NBP requests}
001540    nTQueue: Ptr;                                {pointer to registered name queue}
001541    LAlength: INTEGER;                           {length in bytes of data link addr}
001542    linkAddr: Ptr;                               {data link address returned}
001543    zoneName: Ptr);                              {zone name returned}
001544     ATalkClosePrepParm:
001545    (appName: Ptr);                              {pointer to application name in buffer}
001546   END;
001547
001548  ATPPBPtr = ^ATPParamBlock;
001549  ATPParamBlock = PACKED RECORD
001550   qLink: QElemPtr;                               {next queue entry}
001551   qType: INTEGER;                                {queue type}
001552   ioTrap: INTEGER;                               {routine trap}
001553   ioCmdAddr: Ptr;                                {routine address}
001554   ioCompletion: ProcPtr;                         {completion routine}
001555   ioResult: OSErr;                               {result code}
001556   userData: LONGINT;                             {ATP user bytes}
001557   reqTID: INTEGER;                               {request transaction ID}
```

```
001558   ioRefNum: INTEGER;                          {driver reference number}
001559   csCode: INTEGER;                            {Call command code automatically set}
001560   atpSocket: Byte;                            {currBitMap or socket number}
001561   CASE MPPParmType OF
001562     SendRequestParm, SendResponseParm, GetRequestParm, AddResponseParm, KillSendReqParm:
001563    (atpFlags: Byte;                            {control information}
001564    addrBlock: AddrBlock;                       {source/dest. socket address}
001565    reqLength: INTEGER;                         {request/response length}
001566    reqPointer: Ptr;                            {-> request/response data}
001567    bdsPointer: Ptr;                            {-> response BDS}
001568    CASE MPPParmType OF
001569      SendRequestParm:
001570     (numOfBuffs: Byte;                         {number of responses expected}
001571     timeOutVal: Byte;                          {timeout interval}
001572     numOfResps: Byte;                          {number of responses actually received}
001573     retryCount: Byte;                          {number of retries}
001574     intBuff: INTEGER;                          {used internally for NSendRequest}
001575     TRelTime: Byte);                           {TRelease time for extended send request}
001576      SendResponseParm:
001577     (filler0: Byte;                            {numOfBuffs}
001578     bdsSize: Byte;                             {number of BDS elements}
001579     transID: INTEGER);                         {transaction ID}
001580      GetRequestParm:
001581     (bitMap: Byte;                             {bit map}
001582     filler1: Byte);
001583      AddResponseParm:
001584     (rspNum: Byte;                             {sequence number}
001585     filler2: Byte);
001586      KillSendReqParm:
001587     (aKillQEl: Ptr));                          {ptr to Q element to cancel}
001588   END;
001589
001590  XPPPrmBlkType = (XPPPrmBlk,ASPSizeBlk,ASPAbortPrm,XCallParam);
001591
001592  XPPSubPrmType = (ASPOpenPrm,ASPSubPrm);
001593
001594  XPPEndPrmType = (AFPLoginPrm,ASPEndPrm);
001595
001596  XPPParmBlkPtr = ^XPPParamBlock;
001597  XPPParamBlock = PACKED RECORD
001598   qLink: QElemPtr;                             {next queue entry}
001599   qType: INTEGER;                              {queue type}
001600   ioTrap: INTEGER;                             {routine trap}
001601   ioCmdAddr: Ptr;                              {routine address}
001602   ioCompletion: ProcPtr;                       {completion routine}
001603   ioResult: OSErr;                             {result code}
001604   cmdResult: LONGINT;                          {command result (ATP user bytes)}
001605   ioVRefNum: INTEGER;                          {volume reference or drive number}
001606   ioRefNum: INTEGER;                           {driver reference number}
001607   csCode: INTEGER;                             {call command code}
001608   CASE XPPPrmBlkType OF
001609     ASPAbortPrm:
001610    (abortSCBPtr: Ptr);                         {SCB pointer for AbortOS}
001611     ASPSizeBlk:
001612    (aspMaxCmdSize: INTEGER;                     {for SPGetParms}
001613    aspQuantumSize: INTEGER;                     {for SPGetParms}
```

```
001614    numSesss: INTEGER);                          {for SPGetParms}
001615     XPPPrmBlk:
001616    (sessRefnum: INTEGER;                        {offset to session refnum}
001617    aspTimeout: Byte;                            {timeout for ATP}
001618    aspRetry: Byte;                              {retry count for ATP}
001619    CASE XPPSubPrmType OF
001620      ASPOpenPrm:
001621     (serverAddr: AddrBlock;                     {server address block}
001622     scbPointer: Ptr;                            {SCB pointer}
001623     attnRoutine: Ptr);                          {attention routine pointer}
001624      ASPSubPrm:
001625     (cbSize: INTEGER;                           {command block size}
001626     cbPtr: Ptr;                                 {command block pointer}
001627     rbSize: INTEGER;                            {reply buffer size}
001628     rbPtr: Ptr;                                 {reply buffer pointer}
001629     CASE XPPEndPrmType OF
001630       AFPLoginPrm:
001631      (afpAddrBlock: AddrBlock;                  {address block in AFP login}
001632      afpSCBPtr: Ptr;                            {SCB pointer in AFP login}
001633      afpAttnRoutine: Ptr);                      {Attn routine pointer in AFP login}
001634       ASPEndPrm:
001635      (wdSize: INTEGER;                          {write data size}
001636      wdPtr: Ptr;                                {write data pointer}
001637      ccbStart: ARRAY [0..295] OF Byte)));       {afpWrite max size = 296, else 150}
001638     XCallParam:
001639    (xppSubCode: INTEGER;
001640    xppTimeout: Byte;                            {retry interval (seconds)}
001641    xppRetry: Byte;                              {retry count}
001642    filler1: INTEGER;                            {word space for rent.  see the super.}
001643    zipBuffPtr: Ptr;                             {pointer to buffer (must be 578 bytes)}
001644    zipNumZones: INTEGER;                        {no. of zone names in this response}
001645    zipLastFlag: Byte;                           {non-zero if no more zones}
001646    filler2: Byte;                               {filler}
001647    zipInfoField: PACKED ARRAY [1..70] OF Byte); {on initial call, set first word to zero}
001648   END;
001649
001650
001651  FUNCTION OpenXPP(VAR xppRefnum: INTEGER): OSErr;
001652  FUNCTION ASPOpenSession(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001653  FUNCTION ASPCloseSession(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001654  FUNCTION ASPAbortOS(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001655  FUNCTION ASPGetParms(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001656  FUNCTION ASPCloseAll(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001657  FUNCTION ASPUserWrite(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001658  FUNCTION ASPUserCommand(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001659  FUNCTION ASPGetStatus(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001660  FUNCTION AFPCommand(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001661  FUNCTION GetLocalZones(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001662  FUNCTION GetZoneList(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001663  FUNCTION GetMyZone(thePBptr: XPPParmBlkPtr;async: BOOLEAN): OSErr;
001664  FUNCTION PAttachPH(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001665  FUNCTION PDetachPH(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001666  FUNCTION PWriteLAP(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001667  FUNCTION POpenSkt(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001668  FUNCTION PCloseSkt(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001669  FUNCTION PWriteDDP(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
```

```
001670  FUNCTION PRegisterName(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001671  FUNCTION PLookupName(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001672  FUNCTION PConfirmName(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001673  FUNCTION PRemoveName(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001674  FUNCTION PSetSelfSend(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001675  FUNCTION PKillNBP(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001676  FUNCTION PGetAppleTalkInfo(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001677  FUNCTION PATalkClosePrep(thePBptr: MPPPBPtr;async: BOOLEAN): OSErr;
001678  FUNCTION POpenATPSkt(thePBptr: ATPPBPtr;async: BOOLEAN): OSErr;
001679  FUNCTION PCloseATPSkt(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001680  FUNCTION PSendRequest(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001681  FUNCTION PGetRequest(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001682  FUNCTION PSendResponse(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001683  FUNCTION PAddResponse(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001684  FUNCTION PRelTCB(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001685  FUNCTION PRelRspCB(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001686  FUNCTION PNSendRequest(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001687  FUNCTION PKillSendReq(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001688  FUNCTION PKillGetReq(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001689  FUNCTION ATPKillAllGetReq(thePBPtr: ATPPBPtr;async: BOOLEAN): OSErr;
001690  PROCEDURE BuildLAPwds(wdsPtr: Ptr;dataPtr: Ptr;destHost: INTEGER;prototype: INTEGER;
001691   frameLen: INTEGER);
001692  PROCEDURE BuildDDPwds(wdsPtr: Ptr;headerPtr: Ptr;dataPtr: Ptr;netAddr: AddrBlock;
001693   ddpType: INTEGER;dataLen: INTEGER);
001694  PROCEDURE NBPSetEntity(buffer: Ptr;nbpObject: Str32;nbpType: Str32;nbpZone: Str32);
001695  PROCEDURE NBPSetNTE(ntePtr: Ptr;nbpObject: Str32;nbpType: Str32;nbpZone: Str32;
001696   socket: INTEGER);
001697  FUNCTION GetBridgeAddress: INTEGER;
001698  FUNCTION BuildBDS(buffPtr: Ptr;bdsPtr: Ptr;buffSize: INTEGER): INTEGER;
001699  FUNCTION MPPOpen: OSErr;
001700  FUNCTION MPPClose: OSErr;
001701  FUNCTION LAPOpenProtocol(theLAPType: ABByte;protoPtr: Ptr): OSErr;
001702  FUNCTION LAPCloseProtocol(theLAPType: ABByte): OSErr;
001703  FUNCTION LAPWrite(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001704  FUNCTION LAPRead(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001705  FUNCTION LAPRdCancel(abRecord: ABRecHandle): OSErr;
001706  FUNCTION LAPAddATQ(theATQEntry: ATQEntryPtr): OSErr;
001707  FUNCTION LAPRmvATQ(theATQEntry: ATQEntryPtr): OSErr;
001708  FUNCTION DDPOpenSocket(VAR theSocket: Byte;sktListener: Ptr): OSErr;
001709  FUNCTION DDPCloseSocket(theSocket: Byte): OSErr;
001710  FUNCTION DDPRead(abRecord: ABRecHandle;retCksumErrs: BOOLEAN;async: BOOLEAN): OSErr;
001711  FUNCTION DDPWrite(abRecord: ABRecHandle;doChecksum: BOOLEAN;async: BOOLEAN): OSErr;
001712  FUNCTION DDPRdCancel(abRecord: ABRecHandle): OSErr;
001713  FUNCTION ATPLoad: OSErr;
001714  FUNCTION ATPUnload: OSErr;
001715  FUNCTION ATPOpenSocket(addrRcvd: AddrBlock;VAR atpSocket: Byte): OSErr;
001716  FUNCTION ATPCloseSocket(atpSocket: Byte): OSErr;
001717  FUNCTION ATPSndRequest(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001718  FUNCTION ATPRequest(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001719  FUNCTION ATPReqCancel(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001720  FUNCTION ATPGetRequest(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001721  FUNCTION ATPSndRsp(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001722  FUNCTION ATPAddRsp(abRecord: ABRecHandle): OSErr;
001723  FUNCTION ATPResponse(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001724  FUNCTION ATPRspCancel(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001725  FUNCTION NBPRegister(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
```

```
001726  FUNCTION NBPLookup(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001727  FUNCTION NBPExtract(theBuffer: Ptr;numInBuf: INTEGER;whichOne: INTEGER;
001728   VAR abEntity: EntityName;VAR address: AddrBlock): OSErr;
001729  FUNCTION NBPConfirm(abRecord: ABRecHandle;async: BOOLEAN): OSErr;
001730  FUNCTION NBPRemove(abEntity: EntityPtr): OSErr;
001731  FUNCTION NBPLoad: OSErr;
001732  FUNCTION NBPUnload: OSErr;
001733  FUNCTION GetNodeAddress(VAR myNode: INTEGER;VAR myNet: INTEGER): OSErr;
001734  FUNCTION IsMPPOpen: BOOLEAN;
001735  FUNCTION IsATPOpen: BOOLEAN;
001736  PROCEDURE ATEvent(event: LONGINT;infoPtr: Ptr);
001737  FUNCTION ATPreFlightEvent(event: LONGINT;cancel: LONGINT;infoPtr: Ptr): OSErr;
001738
001739
001740  {$ENDC} { UsingAppleTalk }
001741
001742  {$IFC NOT UsingIncludes}
001743   END.
001744  {$ENDC}
001745
001746
001747  ### END OF FILE AppleTalk.p
001748
```

```
001749
001750  ##########################################################################
001751  ### FILE: Balloons.p
001752  ##########################################################################
001753
001754
001755  {
001756  Created: Sunday, September 15, 1991 at 9:56 PM
001757   Balloons.p
001758   Pascal Interface to the Macintosh Libraries
001759
001760    Copyright Apple Computer, Inc. 1990-1991
001761    All rights reserved
001762  }
001763
001764
001765  {$IFC UNDEFINED UsingIncludes}
001766  {$SETC UsingIncludes := 0}
001767  {$ENDC}
001768
001769  {$IFC NOT UsingIncludes}
001770   UNIT Balloons;
001771   INTERFACE
001772  {$ENDC}
001773
001774  {$IFC UNDEFINED UsingBalloons}
001775  {$SETC UsingBalloons := 1}
001776
001777  {$I+}
001778  {$SETC BalloonsIncludes := UsingIncludes}
001779  {$SETC UsingIncludes := 1}
001780  {$IFC UNDEFINED UsingTypes}
001781  {$I $$Shell(PInterfaces)Types.p}
001782  {$ENDC}
001783  {$IFC UNDEFINED UsingQuickdraw}
001784  {$I $$Shell(PInterfaces)Quickdraw.p}
001785  {$ENDC}
001786  {$IFC UNDEFINED UsingMenus}
001787  {$I $$Shell(PInterfaces)Menus.p}
001788  {$ENDC}
001789  {$IFC UNDEFINED UsingTextEdit}
001790  {$I $$Shell(PInterfaces)TextEdit.p}
001791  {$ENDC}
001792  {$IFC UNDEFINED UsingTraps}
001793  {$I $$Shell(PInterfaces)Traps.p}
001794  {$ENDC}
001795  {$SETC UsingIncludes := BalloonsIncludes}
001796
001797  CONST
001798  hmBalloonHelpVersion = $0002;   { The real version of the Help Manager }
001799
001800  {Help Mgr error range: -850 to -874}
001801  hmHelpDisabled = -850;          { Show Balloons mode was off, call to routine ignored }
001802  hmBalloonAborted = -853;        { Returned if mouse was moving or mouse wasn't in window port rect }
001803  hmSameAsLastBalloon = -854;     { Returned from HMShowMenuBalloon if menu & item is same as last time }
001804  hmHelpManagerNotInited = -855;  { Returned from HMGetHelpMenuHandle if help menu not setup }
```

```
001805  hmSkippedBalloon = -857;        { Returned from calls if helpmsg specified a skip balloon }
001806  hmWrongVersion = -858;          { Returned if help mgr resource was the wrong version }
001807  hmUnknownHelpType = -859;       { Returned if help msg record contained a bad type }
001808  hmOperationUnsupported = -861;  { Returned from HMShowBalloon call if bad method passed to routine }
001809  hmNoBalloonUp = -862;           { Returned from HMRemoveBalloon if no balloon was visible when call was made }
001810  hmCloseViewActive = -863;       { Returned from HMRemoveBalloon if CloseView was active }
001811
001812  kHMHelpMenuID = -16490;         { Resource ID and menu ID of help menu }
001813  kHMAboutHelpItem = 1;           { help menu item number of About Balloon Help… }
001814  kHMShowBalloonsItem = 3;        { help menu item number of Show/Hide Balloons }
001815
001816  kHMHelpID = -5696;              { ID of various Help Mgr package resources (in Pack14 range) }
001817  kBalloonWDEFID = 126;           { Resource ID of the WDEF proc used in standard balloons }
001818
001819  { Dialog item template type constant }
001820  helpItem = 1;                   { key value in DITL template that corresponds to the help item }
001821
001822  { Options for Help Manager resources in 'hmnu', 'hdlg', 'hrct', 'hovr', & 'hfdr' resources }
001823  hmDefaultOptions = 0;           { default options for help manager resources }
001824  hmUseSubID = 1;                 { treat resID's in resources as subID's of driver base ID (for Desk Accessories) }
001825  hmAbsoluteCoords = 2;           { ignore window port origin and treat rectangles as absolute coords (local to window) }
001826  hmSaveBitsNoWindow = 4;         { don't create a window, just blast bits on screen. No update event is generated }
001827  hmSaveBitsWindow = 8;           { create a window, but restore bits behind window when window goes away & generate update event }
001828  hmMatchInTitle = 16;            { for hwin resources, match string anywhere in window title string }
001829
001830  { Constants for Help Types in 'hmnu', 'hdlg', 'hrct', 'hovr', & 'hfdr' resources }
001831  kHMStringItem = 1;              { pstring used in resource }
001832  kHMPictItem = 2;                { 'PICT' ResID used in resource }
001833  kHMStringResItem = 3;           { 'STR#' ResID & index used in resource }
001834  kHMTEResItem = 6;               { Styled Text Edit ResID used in resource ('TEXT' & 'styl') }
001835  kHMSTRResItem = 7;              { 'STR ' ResID used in resource }
001836  kHMSkipItem = 256;              { don't display a balloon }
001837  kHMCompareItem = 512;           { Compare pstring in menu item w/ PString in resource item ('hmnu' only) }
001838  kHMNamedResourceItem = 1024;    { Use pstring in menu item to get 'STR#', 'PICT', or 'STR ' resource ('hmnu' only) }
001839  kHMTrackCntlItem = 2048;        { Reserved }
001840
001841  { Constants for hmmHelpType's when filling out HMMessageRecord }
001842  khmmString = 1;                 { help message contains a PString }
001843  khmmPict = 2;                   { help message contains a resource ID to a 'PICT' resource }
001844  khmmStringRes = 3;              { help message contains a res ID & index to a 'STR#' resource }
001845  khmmTEHandle = 4;               { help message contains a Text Edit handle }
001846  khmmPictHandle = 5;             { help message contains a Picture handle }
001847  khmmTERes = 6;                  { help message contains a res ID to 'TEXT' & 'styl' resources }
001848  khmmSTRRes = 7;                 { help message contains a res ID to a 'STR ' resource }
001849
001850  { ResTypes for Styled TE Handles in Resources }
001851  kHMTETextResType = 'TEXT';      { Resource Type of text data for styled TE record w/o style info }
001852  kHMTEStyleResType = 'styl';     { Resource Type of style information for styled TE record }
001853
001854  { Generic defines for the state parameter used when extracting 'hmnu' & 'hdlg' messages }
001855  kHMEnabledItem = 0;             { item is enabled, but not checked or control value = 0 }
001856  kHMDisabledItem = 1;            { item is disabled, grayed in menus or disabled in dialogs }
001857  kHMCheckedItem = 2;             { item is enabled, and checked or control value = 1 }
001858  kHMOtherItem = 3;               { item is enabled, and control value > 1 }
001859
001860  { Resource Types for whichType parameter used when extracting 'hmnu' & 'hdlg' messages }
```

```
001861  kHMMenuResType = 'hmnu';        { ResType of help resource for supporting menus }
001862  kHMDialogResType = 'hdlg';      { ResType of help resource for supporting dialogs }
001863  kHMWindListResType = 'hwin';    { ResType of help resource for supporting windows }
001864  kHMRectListResType = 'hrct';    { ResType of help resource for rectangles in windows }
001865  kHMOverrideResType = 'hovr';    { ResType of help resource for overriding system balloons }
001866  kHMFinderApplResType = 'hfdr';  { ResType of help resource for custom balloon in Finder }
001867
001868  { Method parameters to pass to HMShowBalloon }
001869  kHMRegularWindow = 0;           { Create a regular window floating above all windows }
001870  kHMSaveBitsNoWindow = 1;        { Just save the bits and draw (for MDEF calls) }
001871  kHMSaveBitsWindow = 2;          { Regular window, save bits behind, AND generate update event }
001872
001873  TYPE
001874  HMStringResType = RECORD
001875   hmmResID: INTEGER;
001876   hmmIndex: INTEGER;
001877   END;
001878
001879  HMMessageRecPtr    = ^HMMessageRecord;
001880  HMMessageRecord    = RECORD
001881                  hmmHelpType            : INTEGER;
001882                  CASE INTEGER OF
001883                  khmmString:
001884                      (hmmString: STR255);
001885                  khmmPict:
001886                      (hmmPict: INTEGER);
001887                  khmmStringRes:
001888                      (hmmStringRes: HMStringResType);
001889                  khmmTEHandle:
001890                      (hmmTEHandle: TEHandle);
001891                  khmmPictHandle:
001892                      (hmmPictHandle: PicHandle);
001893                  khmmTERes:
001894                      (hmmTERes: INTEGER);
001895                  khmmSTRRes:
001896                      (hmmSTRRes: INTEGER);
001897                  END;
001898
001899
001900
001901  {  Public Interfaces  }
001902  FUNCTION HMGetHelpMenuHandle(VAR mh: MenuHandle): OSErr;
001903   INLINE $303C,$0200,_Pack14;
001904  FUNCTION HMShowBalloon(aHelpMsg: HMMessageRecord;
001905                         tip: Point;
001906                         alternateRect: RectPtr;
001907                         tipProc: Ptr;
001908                         theProc: INTEGER;
001909                         variant: INTEGER;
001910                         method: INTEGER): OSErr;
001911   INLINE $303C,$0B01,_Pack14;
001912  FUNCTION HMRemoveBalloon: OSErr;
001913   INLINE $303C,$0002,_Pack14;
001914  FUNCTION HMGetBalloons: BOOLEAN;
001915   INLINE $303C,$0003,_Pack14;
001916  FUNCTION HMSetBalloons(flag: BOOLEAN): OSErr;
```

```
001917   INLINE $303C,$0104,_Pack14;
001918  FUNCTION HMShowMenuBalloon(itemNum: INTEGER;
001919                             itemMenuID: INTEGER;
001920                             itemFlags: LONGINT;
001921                             itemReserved: LONGINT;
001922                             tip: Point;
001923                             alternateRect: RectPtr;
001924                             tipProc: Ptr;
001925                             theProc: INTEGER;
001926                             variant: INTEGER): OSErr;
001927   INLINE $303C,$0E05,_Pack14;
001928  FUNCTION HMGetIndHelpMsg(whichType: ResType;
001929                           whichResID: INTEGER;
001930                           whichMsg: INTEGER;
001931                           whichState: INTEGER;
001932                           VAR options: LONGINT;
001933                           VAR tip: Point;
001934                           VAR altRect: Rect;
001935                           VAR theProc: INTEGER;
001936                           VAR variant: INTEGER;
001937                           VAR aHelpMsg: HMMessageRecord;
001938                           VAR count: INTEGER): OSErr;
001939   INLINE $303C,$1306,_Pack14;
001940  FUNCTION HMIsBalloon: BOOLEAN;
001941   INLINE $303C,$0007,_Pack14;
001942  FUNCTION HMSetFont(font: INTEGER): OSErr;
001943   INLINE $303C,$0108,_Pack14;
001944  FUNCTION HMSetFontSize(fontSize: INTEGER): OSErr;
001945   INLINE $303C,$0109,_Pack14;
001946  FUNCTION HMGetFont(VAR font: INTEGER): OSErr;
001947   INLINE $303C,$020A,_Pack14;
001948  FUNCTION HMGetFontSize(VAR fontSize: INTEGER): OSErr;
001949   INLINE $303C,$020B,_Pack14;
001950  FUNCTION HMSetDialogResID(resID: INTEGER): OSErr;
001951   INLINE $303C,$010C,_Pack14;
001952  FUNCTION HMSetMenuResID(menuID: INTEGER;
001953                          resID: INTEGER): OSErr;
001954   INLINE $303C,$020D,_Pack14;
001955  FUNCTION HMBalloonRect(aHelpMsg: HMMessageRecord;
001956                         VAR coolRect: Rect): OSErr;
001957   INLINE $303C,$040E,_Pack14;
001958  FUNCTION HMBalloonPict(aHelpMsg: HMMessageRecord;
001959                         VAR coolPict: PicHandle): OSErr;
001960   INLINE $303C,$040F,_Pack14;
001961  FUNCTION HMScanTemplateItems(whichID: INTEGER;
001962                               whichResFile: INTEGER;
001963                               whichType: ResType): OSErr;
001964   INLINE $303C,$0410,_Pack14;
001965  FUNCTION HMExtractHelpMsg(whichType: ResType;whichResID: INTEGER;whichMsg: INTEGER;
001966   whichState: INTEGER;VAR aHelpMsg: HMMessageRecord): OSErr;
001967   INLINE $303C,$0711,_Pack14;
001968  FUNCTION HMGetDialogResID(VAR resID: INTEGER): OSErr;
001969   INLINE $303C,$0213,_Pack14;
001970  FUNCTION HMGetMenuResID(menuID: INTEGER;VAR resID: INTEGER): OSErr;
001971   INLINE $303C,$0314,_Pack14;
001972  FUNCTION HMGetBalloonWindow(VAR window: WindowPtr): OSErr;
```

```
001973   INLINE $303C,$0215,_Pack14;
001974
001975
001976  {$ENDC} { UsingBalloons }
001977
001978  {$IFC NOT UsingIncludes}
001979   END.
001980  {$ENDC}
001981
001982
001983  ### END OF FILE Balloons.p
001984
```

```
001985
001986  ############################################################################
001987  ### FILE: CommResources.p
001988  ############################################################################
001989
001990
001991  {
001992  Created: Thursday, September 12, 1991 at 11:54 AM
001993   CommResources.p
001994   Pascal Interface to the Macintosh Libraries
001995
001996    Copyright Apple Computer, Inc. 1988-1991
001997    All rights reserved
001998  }
001999
002000
002001  {$IFC UNDEFINED UsingIncludes}
002002  {$SETC UsingIncludes := 0}
002003  {$ENDC}
002004
002005  {$IFC NOT UsingIncludes}
002006   UNIT CommResources;
002007   INTERFACE
002008  {$ENDC}
002009
002010  {$IFC UNDEFINED UsingCommResources}
002011  {$SETC UsingCommResources := 1}
002012
002013  {$I+}
002014  {$SETC CommResourcesIncludes := UsingIncludes}
002015  {$SETC UsingIncludes := 1}
002016  {$IFC UNDEFINED UsingOSUtils}
002017  {$I $$Shell(PInterfaces)OSUtils.p}
002018  {$ENDC}
002019  {$SETC UsingIncludes := CommResourcesIncludes}
002020
002021  CONST
002022
002023  {    version of the Comm Resource Manager    }
002024  curCRMVersion = 2;
002025
002026  {    tool classes (also the tool file types)    }
002027  classCM = 'cbnd';
002028  classFT = 'fbnd';
002029  classTM = 'tbnd';
002030
002031  { constants general to the use of the Communications Resource Manager }
002032  crmType = 9;                { queue type    }
002033  crmRecVersion = 1;          { version of queue structure }
002034
002035  {    error codes }
002036  crmGenericError = -1;
002037  crmNoErr = 0;
002038
002039  TYPE
002040  { data structures general to the use of the Communications Resource Manager }
```

```
002041  CRMErr = OSErr;
002042
002043  CRMRecPtr = ^CRMRec;
002044  CRMRec = RECORD
002045   qLink: QElemPtr;           {reserved}
002046   qType: INTEGER;            {queue type -- ORD(crmType) = 9}
002047   crmVersion: INTEGER;       {version of queue element data structure}
002048   crmPrivate: LONGINT;       {reserved}
002049   crmReserved: INTEGER;      {reserved}
002050   crmDeviceType: LONGINT;    {type of device, assigned by DTS}
002051   crmDeviceID: LONGINT;      {device ID; assigned when CRMInstall is called}
002052   crmAttributes: LONGINT;    {pointer to attribute block}
002053   crmStatus: LONGINT;        {status variable - device specific}
002054   crmRefCon: LONGINT;        {for device private use}
002055   END;
002056
002057
002058  FUNCTION InitCRM: CRMErr;
002059  FUNCTION CRMGetHeader: QHdrPtr;
002060  PROCEDURE CRMInstall(crmReqPtr: QElemPtr);
002061  FUNCTION CRMRemove(crmReqPtr: QElemPtr): OSErr;
002062  FUNCTION CRMSearch(crmReqPtr: QElemPtr): QElemPtr;
002063  FUNCTION CRMGetCRMVersion: INTEGER;
002064
002065  FUNCTION CRMGetResource(theType: ResType;theID: INTEGER): Handle;
002066  FUNCTION CRMGet1Resource(theType: ResType;theID: INTEGER): Handle;
002067  FUNCTION CRMGetIndResource(theType: ResType;index: INTEGER): Handle;
002068  FUNCTION CRMGet1IndResource(theType: ResType;index: INTEGER): Handle;
002069  FUNCTION CRMGetNamedResource(theType: ResType;name: Str255): Handle;
002070  FUNCTION CRMGet1NamedResource(theType: ResType;name: Str255): Handle;
002071  PROCEDURE CRMReleaseResource(theHandle: Handle);
002072  FUNCTION CRMGetToolResource(procID: INTEGER;theType: ResType;theID: INTEGER): Handle;
002073  FUNCTION CRMGetToolNamedResource(procID: INTEGER;theType: ResType;name: Str255): Handle;
002074  PROCEDURE CRMReleaseToolResource(procID: INTEGER;theHandle: Handle);
002075  FUNCTION CRMGetIndex(theHandle: Handle): LONGINT;
002076
002077  FUNCTION CRMLocalToRealID(bundleType: ResType;toolID: INTEGER;theType: ResType;
002078   localID: INTEGER): INTEGER;
002079  FUNCTION CRMRealToLocalID(bundleType: ResType;toolID: INTEGER;theType: ResType;
002080   realID: INTEGER): INTEGER;
002081
002082  FUNCTION CRMGetIndToolName(bundleType: OSType;index: INTEGER;VAR toolName: Str255): OSErr;
002083
002084  FUNCTION CRMFindCommunications(VAR vRefNum: INTEGER;VAR dirID: LONGINT): OSErr;
002085
002086  FUNCTION CRMIsDriverOpen(driverName: Str255): BOOLEAN;
002087
002088  FUNCTION CRMParseCAPSResource(theHandle: Handle;selector: ResType;VAR value: LONGINT): CRMErr;
002089
002090  FUNCTION CRMReserveRF(refNum: INTEGER): OSErr;
002091
002092  {  decrements useCount by one  }
002093  FUNCTION CRMReleaseRF(refNum: INTEGER): OSErr;
002094
002095
002096
```

```
002097  {$ENDC} { UsingCommResources }
002098
002099  {$IFC NOT UsingIncludes}
002100   END.
002101  {$ENDC}
002102
002103
002104  ### END OF FILE CommResources.p
002105
```

```
002106
002107  ############################################################################
002108  ### FILE: Connections.p
002109  ############################################################################
002110
002111
002112  {
002113  Created: Wednesday, September 11, 1991 at 11:34 AM
002114   Connections.p
002115   Pascal Interface to the Macintosh Libraries
002116
002117    Copyright Apple Computer, Inc. 1988-1991
002118    All rights reserved
002119  }
002120
002121
002122  {$IFC UNDEFINED UsingIncludes}
002123  {$SETC UsingIncludes := 0}
002124  {$ENDC}
002125
002126  {$IFC NOT UsingIncludes}
002127   UNIT Connections;
002128   INTERFACE
002129  {$ENDC}
002130
002131  {$IFC UNDEFINED UsingConnections}
002132  {$SETC UsingConnections := 1}
002133
002134  {$I+}
002135  {$SETC ConnectionsIncludes := UsingIncludes}
002136  {$SETC UsingIncludes := 1}
002137  {$IFC UNDEFINED UsingDialogs}
002138  {$I $$Shell(PInterfaces)Dialogs.p}
002139  {$ENDC}
002140  {$IFC UNDEFINED UsingCTBUtilities}
002141  {$I $$Shell(PInterfaces)CTBUtilities.p}
002142  {$ENDC}
002143  {$SETC UsingIncludes := ConnectionsIncludes}
002144
002145  CONST
002146
002147  {    current Connection Manager version    }
002148  curCMVersion = 2;
002149
002150  {    current Connection Manager Environment Record version     }
002151  curConnEnvRecVers = 0;
002152
002153  { CMErr }
002154  cmGenericError = -1;
002155  cmNoErr = 0;
002156  cmRejected = 1;
002157  cmFailed = 2;
002158  cmTimeOut = 3;
002159  cmNotOpen = 4;
002160  cmNotClosed = 5;
002161  cmNoRequestPending = 6;
```

```
002162  cmNotSupported = 7;
002163  cmNoTools = 8;
002164  cmUserCancel = 9;
002165  cmUnknownError = 11;
002166
002167  TYPE
002168  CMErr = OSErr;
002169
002170  CONST
002171
002172  { Low word of CMRecFlags is same as CMChannel }
002173  cmData = $00000001;
002174  cmCntl = $00000002;
002175  cmAttn = $00000004;
002176  cmDataNoTimeout = $00000010;
002177  cmCntlNoTimeout = $00000020;
002178  cmAttnNoTimeout = $00000040;
002179  cmDataClean = $00000100;
002180  cmCntlClean = $00000200;
002181  cmAttnClean = $00000400;
002182
002183  { only for CMRecFlags (not CMChannel) below this point }
002184  cmNoMenus = $00010000;
002185  cmQuiet = $00020000;
002186  cmConfigChanged = $00040000;
002187
002188  TYPE
002189  CMRecFlags = LONGINT;
002190
002191
002192  CMChannel = INTEGER;
002193
002194  CONST
002195
002196  { CMStatFlags }
002197  cmStatusOpening = $00000001;
002198  cmStatusOpen = $00000002;
002199  cmStatusClosing = $00000004;
002200  cmStatusDataAvail = $00000008;
002201  cmStatusCntlAvail = $00000010;
002202  cmStatusAttnAvail = $00000020;
002203  cmStatusDRPend = $00000040;     {data read pending}
002204  cmStatusDWPend = $00000080;     {data write pending}
002205  cmStatusCRPend = $00000100;     {cntl read pending}
002206  cmStatusCWPend = $00000200;     {cntl write pending}
002207  cmStatusARPend = $00000400;     {attn read pending}
002208  cmStatusAWPend = $00000800;     {attn write pending}
002209  cmStatusBreakPend = $00001000;
002210  cmStatusListenPend = $00002000;
002211  cmStatusIncomingCallPresent = $00004000;
002212  cmStatusReserved0 = $00008000;
002213
002214  TYPE
002215  CMStatFlags = LONGINT;
002216
002217  CMBufFields = (cmDataIn,cmDataOut,cmCntlIn,cmCntlOut,cmAttnIn,cmAttnOut,
```

```
002218   cmRsrvIn,cmRsrvOut);
002219
002220
002221  CMBuffers = ARRAY[CMBufFields] OF Ptr;
002222  CMBufferSizes = ARRAY[CMBufFields] OF LONGINT;
002223
002224  CONST
002225
002226  { CMSearchFlags }
002227  cmSearchSevenBit = $0001;
002228
002229  TYPE
002230  CMSearchFlags = INTEGER;
002231
002232
002233  CONST
002234
002235  { CMFlags }
002236  cmFlagsEOM = $0001;
002237
002238  TYPE
002239  CMFlags = INTEGER;
002240
002241
002242  ConnEnvironRecPtr = ^ConnEnvironRec;
002243  ConnEnvironRec = RECORD
002244   version: INTEGER;
002245   baudRate: LONGINT;
002246   dataBits: INTEGER;
002247   channels: CMChannel;
002248   swFlowControl: BOOLEAN;
002249   hwFlowControl: BOOLEAN;
002250   flags: CMFlags;
002251   END;
002252
002253  ConnPtr = ^ConnRecord;
002254  ConnHandle = ^ConnPtr;
002255  ConnRecord = RECORD
002256   procID: INTEGER;
002257   flags: CMRecFlags;
002258   errCode: CMErr;
002259   refCon: LONGINT;
002260   userData: LONGINT;
002261   defProc: ProcPtr;
002262   config: Ptr;
002263   oldConfig: Ptr;
002264   asyncEOM: LONGINT;
002265   reserved1: LONGINT;
002266   reserved2: LONGINT;
002267   cmPrivate: Ptr;
002268   bufferArray: CMBuffers;
002269   bufSizes: CMBufferSizes;
002270   mluField: LONGINT;
002271   asyncCount: CMBufferSizes;
002272   END;
002273
```

```
002274
002275  { application routines type definitions }
002276  ConnectionSearchCallBackProcPtr = ProcPtr;
002277  ConnectionCompletionProcPtr = ProcPtr;
002278  ConnectionChooseIdleProcPtr = ProcPtr;
002279
002280  CONST
002281
002282  { CMIOPB constants and structure }
002283  cmIOPBQType = 10;
002284  cmIOPBversion = 0;
002285
002286  TYPE
002287  CMIOPBPtr = ^CMIOPB;
002288  CMIOPB = RECORD
002289   qLink: QElemPtr;
002290   qType: INTEGER;                { cmIOPBQType }
002291   hConn: ConnHandle;
002292   theBuffer: Ptr;
002293   count: LONGINT;
002294   flags: CMFlags;
002295   userCompletion: ConnectionCompletionProcPtr;
002296   timeout: LONGINT;
002297   errCode: CMErr;
002298   channel: CMChannel;
002299   asyncEOM: LONGINT;
002300   reserved1: LONGINT;
002301   reserved2: INTEGER;
002302   version: INTEGER;              { cmIOPBversion }
002303   refCon: LONGINT;               { for application }
002304   toolData1: LONGINT;            { for tool }
002305   toolData2: LONGINT;            { for tool }
002306   END;
002307
002308
002309  FUNCTION InitCM: CMErr;
002310  FUNCTION CMGetVersion(hConn: ConnHandle): Handle;
002311  FUNCTION CMGetCMVersion: INTEGER;
002312
002313  FUNCTION CMNew(procID: INTEGER;flags: CMRecFlags;desiredSizes: CMBufferSizes;
002314   refCon: LONGINT;userData: LONGINT): ConnHandle;
002315
002316  PROCEDURE CMDispose(hConn: ConnHandle);
002317
002318  FUNCTION CMListen(hConn: ConnHandle;async: BOOLEAN;completor: ConnectionCompletionProcPtr;
002319   timeout: LONGINT): CMErr;
002320  FUNCTION CMAccept(hConn: ConnHandle;accept: BOOLEAN): CMErr;
002321
002322  FUNCTION CMOpen(hConn: ConnHandle;async: BOOLEAN;completor: ConnectionCompletionProcPtr;
002323   timeout: LONGINT): CMErr;
002324  FUNCTION CMClose(hConn: ConnHandle;async: BOOLEAN;completor: ConnectionCompletionProcPtr;
002325   timeout: LONGINT;now: BOOLEAN): CMErr;
002326
002327  FUNCTION CMAbort(hConn: ConnHandle): CMErr;
002328
002329  FUNCTION CMStatus(hConn: ConnHandle;VAR sizes: CMBufferSizes;VAR flags: CMStatFlags): CMErr;
```

```
002330  PROCEDURE CMIdle(hConn: ConnHandle);
002331
002332  PROCEDURE CMReset(hConn: ConnHandle);
002333
002334  PROCEDURE CMBreak(hConn: ConnHandle;duration: LONGINT;async: BOOLEAN;completor: ConnectionCompletionProcPtr);
002335
002336  FUNCTION CMRead(hConn: ConnHandle;theBuffer: Ptr;VAR toRead: LONGINT;theChannel: CMChannel;
002337   async: BOOLEAN;completor: ConnectionCompletionProcPtr;timeout: LONGINT;
002338   VAR flags: CMFlags): CMErr;
002339  FUNCTION CMWrite(hConn: ConnHandle;theBuffer: Ptr;VAR toWrite: LONGINT;
002340   theChannel: CMChannel;async: BOOLEAN;completor: ConnectionCompletionProcPtr;
002341   timeout: LONGINT;flags: CMFlags): CMErr;
002342
002343  FUNCTION CMIOKill(hConn: ConnHandle;which: INTEGER): CMErr;
002344
002345  PROCEDURE CMActivate(hConn: ConnHandle;activate: BOOLEAN);
002346  PROCEDURE CMResume(hConn: ConnHandle;resume: BOOLEAN);
002347  FUNCTION CMMenu(hConn: ConnHandle;menuID: INTEGER;item: INTEGER): BOOLEAN;
002348
002349  FUNCTION CMValidate(hConn: ConnHandle): BOOLEAN;
002350  PROCEDURE CMDefault(VAR theConfig: Ptr;procID: INTEGER;allocate: BOOLEAN);
002351
002352  FUNCTION CMSetupPreflight(procID: INTEGER;VAR magicCookie: LONGINT): Handle;
002353  FUNCTION CMSetupFilter(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
002354   VAR theEvent: EventRecord;VAR theItem: INTEGER;VAR magicCookie: LONGINT): BOOLEAN;
002355  PROCEDURE CMSetupSetup(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
002356   VAR magicCookie: LONGINT);
002357  PROCEDURE CMSetupItem(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
002358   VAR theItem: INTEGER;VAR magicCookie: LONGINT);
002359  PROCEDURE CMSetupXCleanup(procID: INTEGER;theConfig: Ptr;count: INTEGER;
002360   theDialog: DialogPtr;OKed: BOOLEAN;VAR magicCookie: LONGINT);
002361  PROCEDURE CMSetupPostflight(procID: INTEGER);
002362
002363  FUNCTION CMGetConfig(hConn: ConnHandle): Ptr;
002364  FUNCTION CMSetConfig(hConn: ConnHandle;thePtr: Ptr): INTEGER;
002365
002366  FUNCTION CMIntlToEnglish(hConn: ConnHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
002367   language: INTEGER): OSErr;
002368  FUNCTION CMEnglishToIntl(hConn: ConnHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
002369   language: INTEGER): OSErr;
002370
002371  FUNCTION CMAddSearch(hConn: ConnHandle;theString: Str255;flags: CMSearchFlags;
002372   callBack: ConnectionSearchCallBackProcPtr): LONGINT;
002373  PROCEDURE CMRemoveSearch(hConn: ConnHandle;refnum: LONGINT);
002374  PROCEDURE CMClearSearch(hConn: ConnHandle);
002375
002376  FUNCTION CMGetConnEnvirons(hConn: ConnHandle;VAR theEnvirons: ConnEnvironRec): CMErr;
002377
002378  FUNCTION CMChoose(VAR hConn: ConnHandle;where: Point;idleProc: ConnectionChooseIdleProcPtr): INTEGER;
002379
002380  PROCEDURE CMEvent(hConn: ConnHandle;theEvent: EventRecord);
002381
002382  PROCEDURE CMGetToolName(procID: INTEGER;VAR name: Str255);
002383  FUNCTION CMGetProcID(name: Str255): INTEGER;
002384
002385  PROCEDURE CMSetRefCon(hConn: ConnHandle;refCon: LONGINT);
```

```
002386  FUNCTION CMGetRefCon(hConn: ConnHandle): LONGINT;
002387
002388  FUNCTION CMGetUserData(hConn: ConnHandle): LONGINT;
002389  PROCEDURE CMSetUserData(hConn: ConnHandle;userData: LONGINT);
002390
002391  PROCEDURE CMGetErrorString(hConn: ConnHandle;id: INTEGER;VAR errMsg: Str255);
002392
002393  FUNCTION CMNewIOPB(hConn: ConnHandle;VAR theIOPB: CMIOPBPtr): CMErr;
002394  FUNCTION CMDisposeIOPB(hConn: ConnHandle;theIOPB: CMIOPBPtr): CMErr;
002395
002396  FUNCTION CMPBRead(hConn: ConnHandle;theIOPB: CMIOPBPtr;async: BOOLEAN): CMErr;
002397  FUNCTION CMPBWrite(hConn: ConnHandle;theIOPB: CMIOPBPtr;async: BOOLEAN): CMErr;
002398  FUNCTION CMPBIOKill(hConn: ConnHandle;theIOPB: CMIOPBPtr): CMErr;
002399
002400
002401  {$ENDC} { UsingConnections }
002402
002403  {$IFC NOT UsingIncludes}
002404   END.
002405  {$ENDC}
002406
002407
002408  ### END OF FILE Connections.p
002409
```

```
002410
002411  ############################################################################
002412  ### FILE: ConnectionTools.p
002413  ############################################################################
002414
002415
002416  {
002417  Created: Wednesday, September 11, 1991 at 1:48 PM
002418   ConnectionTools.p
002419   Pascal Interface to the Macintosh Libraries
002420
002421    Copyright Apple Computer, Inc. 1988-1991
002422    All rights reserved
002423  }
002424
002425
002426  {$IFC UNDEFINED UsingIncludes}
002427  {$SETC UsingIncludes := 0}
002428  {$ENDC}
002429
002430  {$IFC NOT UsingIncludes}
002431   UNIT ConnectionTools;
002432   INTERFACE
002433  {$ENDC}
002434
002435  {$IFC UNDEFINED UsingConnectionTools}
002436  {$SETC UsingConnectionTools := 1}
002437
002438  {$I+}
002439  {$SETC ConnectionToolsIncludes := UsingIncludes}
002440  {$SETC UsingIncludes := 1}
002441  {$IFC UNDEFINED UsingDialogs}
002442  {$I $$Shell(PInterfaces)Dialogs.p}
002443  {$ENDC}
002444  {$IFC UNDEFINED UsingConnections}
002445  {$I $$Shell(PInterfaces)Connections.p}
002446  {$ENDC}
002447  {$SETC UsingIncludes := ConnectionToolsIncludes}
002448
002449  CONST
002450
002451  { messages for DefProc }
002452  cmInitMsg = 0;
002453  cmDisposeMsg = 1;
002454  cmSuspendMsg = 2;
002455  cmResumeMsg = 3;
002456  cmMenuMsg = 4;
002457  cmEventMsg = 5;
002458  cmActivateMsg = 6;
002459  cmDeactivateMsg = 7;
002460
002461  cmIdleMsg = 50;
002462  cmResetMsg = 51;
002463  cmAbortMsg = 52;
002464
002465  cmReadMsg = 100;
```

```
002466  cmWriteMsg = 101;
002467  cmStatusMsg = 102;
002468  cmListenMsg = 103;
002469  cmAcceptMsg = 104;
002470  cmCloseMsg = 105;
002471  cmOpenMsg = 106;
002472  cmBreakMsg = 107;
002473  cmIOKillMsg = 108;
002474  cmEnvironsMsg = 109;
002475
002476  { new connection tool messages for ctb 1.1 }
002477  cmNewIOPBMsg = 110;
002478  cmDisposeIOPBMsg = 111;
002479  cmGetErrorStringMsg = 112;
002480  cmPBReadMsg = 113;
002481  cmPBWriteMsg = 114;
002482  cmPBIOKillMsg = 115;
002483
002484  {    messages for validate DefProc    }
002485  cmValidateMsg = 0;
002486  cmDefaultMsg = 1;
002487
002488  {    messages for Setup DefProc    }
002489  cmSpreflightMsg = 0;
002490  cmSsetupMsg = 1;
002491  cmSitemMsg = 2;
002492  cmSfilterMsg = 3;
002493  cmScleanupMsg = 4;
002494
002495  {    messages for scripting defProc    }
002496  cmMgetMsg = 0;
002497  cmMsetMsg = 1;
002498
002499  {    messages for localization defProc    }
002500  cmL2English = 0;
002501  cmL2Intl = 1;
002502
002503  {    private data constants }
002504  cdefType = 'cdef';  { main connection definition procedure    }
002505  cvalType = 'cval';  { validation definition procedure    }
002506  csetType = 'cset';  { connection setup definition procedure    }
002507  clocType = 'cloc';  { connection configuration localization defProc    }
002508  cscrType = 'cscr';  { connection scripting defProc interfaces    }
002509
002510  cbndType = 'cbnd';  { bundle type for connection    }
002511  cverType = 'vers';
002512
002513  TYPE
002514  CMDataBufferPtr = ^CMDataBuffer;
002515  CMDataBuffer = RECORD
002516   thePtr: Ptr;
002517   count: LONGINT;
002518   channel: CMChannel;
002519   flags: CMFlags;
002520   END;
002521
```

```
002522  CMCompletorPtr = ^CMCompletorRecord;
002523  CMCompletorRecord = RECORD
002524   async: BOOLEAN;
002525   completionRoutine: ProcPtr;
002526   END;
002527
002528  { Private Data Structure }
002529
002530  CMSetupPtr = ^CMSetupStruct;
002531  CMSetupStruct = RECORD
002532   theDialog: DialogPtr;
002533   count: INTEGER;
002534   theConfig: Ptr;
002535   procID: INTEGER;   { procID of the tool }
002536   END;
002537
002538
002539
002540  {$ENDC} { UsingConnectionTools }
002541
002542  {$IFC NOT UsingIncludes}
002543   END.
002544  {$ENDC}
002545
002546
002547  ### END OF FILE ConnectionTools.p
002548
```

```
002549
002550  ############################################################################
002551  ### FILE: Controls.p
002552  ############################################################################
002553
002554  {
002555  Created: Sunday, January 6, 1991 at 10:25 PM
002556      Controls.p
002557      Pascal Interface to the Macintosh Libraries
002558
002559          Copyright Apple Computer, Inc. 1985-1990
002560          All rights reserved
002561  }
002562
002563
002564  {$IFC UNDEFINED UsingIncludes}
002565  {$SETC UsingIncludes := 0}
002566  {$ENDC}
002567
002568  {$IFC NOT UsingIncludes}
002569      UNIT Controls;
002570      INTERFACE
002571  {$ENDC}
002572
002573  {$IFC UNDEFINED UsingControls}
002574  {$SETC UsingControls := 1}
002575
002576  {$I+}
002577  {$SETC ControlsIncludes := UsingIncludes}
002578  {$SETC UsingIncludes := 1}
002579  {$IFC UNDEFINED UsingQuickdraw}
002580  {$I $$Shell(PInterfaces)Quickdraw.p}
002581  {$ENDC}
002582  {$SETC UsingIncludes := ControlsIncludes}
002583
002584  CONST
002585  pushButProc = 0;
002586  checkBoxProc = 1;
002587  radioButProc = 2;
002588  useWFont = 8;
002589  scrollBarProc = 16;
002590  inButton = 10;
002591  inCheckBox = 11;
002592  inUpButton = 20;
002593  inDownButton = 21;
002594  inPageUp = 22;
002595  inPageDown = 23;
002596  inThumb = 129;
002597
002598  popupMenuProc = 1008;               { 63 * 16 }
002599  popupFixedWidth = $0001;            { popup menu CDEF variation codes }
002600  popupUseAddResMenu = $0004;
002601  popupUseWFont = $0008;
002602  popupTitleBold = $00000100;         {    Popup Title characteristics }
002603  popupTitleItalic = $00000200;
002604  popupTitleUnderline = $00000400;
```

```
002605  popupTitleOutline = $00000800;
002606  popupTitleShadow = $00001000;
002607  popupTitleCondense = $00002000;
002608  popupTitleExtend = $00004000;
002609  popupTitleNoStyle = $00008000;
002610  popupTitleLeftJust = $00000000;
002611  popupTitleCenterJust = $00000001;
002612  popupTitleRightJust = $000000FF;
002613
002614  {
002615  axis constraints for DragGrayRgn call}
002616  noConstraint = 0;
002617  hAxisOnly = 1;
002618  vAxisOnly = 2;
002619
002620  {
002621  control messages}
002622  drawCntl = 0;
002623  testCntl = 1;
002624  calcCRgns = 2;
002625  initCntl = 3;
002626  dispCntl = 4;
002627  posCntl = 5;
002628  thumbCntl = 6;
002629  dragCntl = 7;
002630  autoTrack = 8;
002631  calcCntlRgn = 10;
002632  calcThumbRgn = 11;
002633
002634  cFrameColor = 0;
002635  cBodyColor = 1;
002636  cTextColor = 2;
002637  cThumbColor = 3;
002638  popupMenuCDEFproc = popupMenuProc;  { synonym for compatibility }
002639
002640  TYPE
002641  ControlPtr = ^ControlRecord;
002642  ControlHandle = ^ControlPtr;
002643  ControlRecord = PACKED RECORD
002644      nextControl: ControlHandle;
002645      contrlOwner: WindowPtr;
002646      contrlRect: Rect;
002647      contrlVis: Byte;
002648      contrlHilite: Byte;
002649      contrlValue: INTEGER;
002650      contrlMin: INTEGER;
002651      contrlMax: INTEGER;
002652      contrlDefProc: Handle;
002653      contrlData: Handle;
002654      contrlAction: ProcPtr;
002655      contrlRfCon: LONGINT;
002656      contrlTitle: Str255;
002657      END;
002658
002659  CCTabPtr = ^CtlCTab;
002660  CCTabHandle = ^CCTabPtr;
```

```
002661  CtlCTab = RECORD
002662      ccSeed: LONGINT;                {reserved}
002663      ccRider: INTEGER;               {see what you have done - reserved}
002664      ctSize: INTEGER;                {usually 3 for controls}
002665      ctTable: ARRAY [0..3] OF ColorSpec;
002666      END;
002667
002668  AuxCtlPtr = ^AuxCtlRec;
002669  AuxCtlHandle = ^AuxCtlPtr;
002670  AuxCtlRec = RECORD
002671      acNext: AuxCtlHandle;           {handle to next AuxCtlRec}
002672      acOwner: ControlHandle;         {handle for aux record's control}
002673      acCTable: CCTabHandle;          {color table for this control}
002674      acFlags: INTEGER;               {misc flag byte}
002675      acReserved: LONGINT;            {reserved for use by Apple}
002676      acRefCon: LONGINT;              {for use by application}
002677      END;
002678
002679
002680  FUNCTION NewControl(theWindow: WindowPtr;boundsRect: Rect;title: Str255;
002681      visible: BOOLEAN;value: INTEGER;min: INTEGER;max: INTEGER;procID: INTEGER;
002682      refCon: LONGINT): ControlHandle;
002683      INLINE $A954;
002684  PROCEDURE SetCTitle(theControl: ControlHandle;title: Str255);
002685      INLINE $A95F;
002686  PROCEDURE GetCTitle(theControl: ControlHandle;VAR title: Str255);
002687      INLINE $A95E;
002688  FUNCTION GetNewControl(controlID: INTEGER;owner: WindowPtr): ControlHandle;
002689      INLINE $A9BE;
002690  PROCEDURE DisposeControl(theControl: ControlHandle);
002691      INLINE $A955;
002692  PROCEDURE KillControls(theWindow: WindowPtr);
002693      INLINE $A956;
002694  PROCEDURE HideControl(theControl: ControlHandle);
002695      INLINE $A958;
002696  PROCEDURE ShowControl(theControl: ControlHandle);
002697      INLINE $A957;
002698  PROCEDURE DrawControls(theWindow: WindowPtr);
002699      INLINE $A969;
002700  PROCEDURE Draw1Control(theControl: ControlHandle);
002701      INLINE $A96D;
002702  PROCEDURE HiliteControl(theControl: ControlHandle;hiliteState: INTEGER);
002703      INLINE $A95D;
002704  PROCEDURE UpdtControl(theWindow: WindowPtr;updateRgn: RgnHandle);
002705      INLINE $A953;
002706  PROCEDURE UpdateControls(theWindow: WindowPtr;updateRgn: RgnHandle);
002707      INLINE $A953;
002708  PROCEDURE MoveControl(theControl: ControlHandle;h: INTEGER;v: INTEGER);
002709      INLINE $A959;
002710  PROCEDURE SizeControl(theControl: ControlHandle;w: INTEGER;h: INTEGER);
002711      INLINE $A95C;
002712  PROCEDURE SetCtlValue(theControl: ControlHandle;theValue: INTEGER);
002713      INLINE $A963;
002714  FUNCTION GetCtlValue(theControl: ControlHandle): INTEGER;
002715      INLINE $A960;
002716  PROCEDURE SetCtlMin(theControl: ControlHandle;minValue: INTEGER);
```

```
002717      INLINE $A964;
002718  FUNCTION GetCtlMin(theControl: ControlHandle): INTEGER;
002719      INLINE $A961;
002720  PROCEDURE SetCtlMax(theControl: ControlHandle;maxValue: INTEGER);
002721      INLINE $A965;
002722  FUNCTION GetCtlMax(theControl: ControlHandle): INTEGER;
002723      INLINE $A962;
002724  PROCEDURE SetCRefCon(theControl: ControlHandle;data: LONGINT);
002725      INLINE $A95B;
002726  FUNCTION GetCRefCon(theControl: ControlHandle): LONGINT;
002727      INLINE $A95A;
002728  PROCEDURE SetCtlAction(theControl: ControlHandle;actionProc: ProcPtr);
002729      INLINE $A96B;
002730  FUNCTION GetCtlAction(theControl: ControlHandle): ProcPtr;
002731      INLINE $A96A;
002732  PROCEDURE DragControl(theControl: ControlHandle;startPt: Point;limitRect: Rect;
002733      slopRect: Rect;axis: INTEGER);
002734      INLINE $A967;
002735  FUNCTION TestControl(theControl: ControlHandle;thePt: Point): INTEGER;
002736      INLINE $A966;
002737  FUNCTION TrackControl(theControl: ControlHandle;thePoint: Point;actionProc: ProcPtr): INTEGER;
002738      INLINE $A968;
002739  FUNCTION FindControl(thePoint: Point;theWindow: WindowPtr;VAR theControl: ControlHandle): INTEGER;
002740      INLINE $A96C;
002741  PROCEDURE SetCtlColor(theControl: ControlHandle;newColorTable: CCTabHandle);
002742      INLINE $AA43;
002743  FUNCTION GetAuxCtl(theControl: ControlHandle;VAR acHndl: AuxCtlHandle): BOOLEAN;
002744      INLINE $AA44;
002745  FUNCTION GetCVariant(theControl: ControlHandle): INTEGER;
002746      INLINE $A809;
002747
002748
002749  {$ENDC}    { UsingControls }
002750
002751  {$IFC NOT UsingIncludes}
002752      END.
002753  {$ENDC}
002754
002755
002756  ### END OF FILE Controls.p
002757
```

```
002758
002759  ############################################################################
002760  ### FILE: CRMSerialDevices.p
002761  ############################################################################
002762
002763
002764  {
002765  Created: Wednesday, September 11, 1991 at 2:05 PM
002766   CRMSerialDevices.p
002767   Pascal Interface to the Macintosh Libraries
002768
002769    Copyright Apple Computer, Inc. 1988-1991
002770    All rights reserved
002771  }
002772
002773
002774  {$IFC UNDEFINED UsingIncludes}
002775  {$SETC UsingIncludes := 0}
002776  {$ENDC}
002777
002778  {$IFC NOT UsingIncludes}
002779   UNIT CRMSerialDevices;
002780   INTERFACE
002781  {$ENDC}
002782
002783  {$IFC UNDEFINED UsingCRMSerialDevices}
002784  {$SETC UsingCRMSerialDevices := 1}
002785
002786  {$I+}
002787  {$SETC CRMSerialDevicesIncludes := UsingIncludes}
002788  {$SETC UsingIncludes := 1}
002789  {$IFC UNDEFINED UsingTypes}
002790  {$I $$Shell(PInterfaces)Types.p}
002791  {$ENDC}
002792  {$SETC UsingIncludes := CRMSerialDevicesIncludes}
002793
002794  CONST
002795
002796  {    for the crmDeviceType field of the CRMRec data structure }
002797  crmSerialDevice = 1;
002798
002799
002800  {  version of the CRMSerialRecord below }
002801  curCRMSerRecVers = 1;
002802
002803  TYPE
002804  { Maintains compatibility w/ apps & tools that expect an old style icon    }
002805  CRMIconPtr = ^CRMIconRecord;
002806  CRMIconHandle = ^CRMIconPtr;
002807  CRMIconRecord = RECORD
002808   oldIcon: ARRAY [0..31] OF LONGINT;     { ICN#    }
002809   oldMask: ARRAY [0..31] OF LONGINT;
002810   theSuite: Handle;                      { Handle to an IconSuite    }
002811   reserved: LONGINT;
002812   END;
002813
```

```
002814  CRMSerialPtr = ^CRMSerialRecord;
002815  CRMSerialRecord = RECORD
002816   version: INTEGER;
002817   inputDriverName: StringHandle;
002818   outputDriverName: StringHandle;
002819   name: StringHandle;
002820   deviceIcon: CRMIconHandle;
002821   ratedSpeed: LONGINT;
002822   maxSpeed: LONGINT;
002823   reserved: LONGINT;
002824   END;
002825
002826
002827
002828  {$ENDC} { UsingCRMSerialDevices }
002829
002830  {$IFC NOT UsingIncludes}
002831   END.
002832  {$ENDC}
002833
002834
002835  ### END OF FILE CRMSerialDevices.p
002836
```

```
002837
002838  ############################################################################
002839  ### FILE: CTBUtilities.p
002840  ############################################################################
002841
002842
002843  {
002844  Created: Thursday, September 12, 1991 at 2:53 PM
002845   CTBUtilities.p
002846   Pascal Interface to the Macintosh Libraries
002847
002848    Copyright Apple Computer, Inc. 1988-1991
002849    All rights reserved
002850  }
002851
002852
002853  {$IFC UNDEFINED UsingIncludes}
002854  {$SETC UsingIncludes := 0}
002855  {$ENDC}
002856
002857  {$IFC NOT UsingIncludes}
002858   UNIT CTBUtilities;
002859   INTERFACE
002860  {$ENDC}
002861
002862  {$IFC UNDEFINED UsingCTBUtilities}
002863  {$SETC UsingCTBUtilities := 1}
002864
002865  {$I+}
002866  {$SETC CTBUtilitiesIncludes := UsingIncludes}
002867  {$SETC UsingIncludes := 1}
002868  {$IFC UNDEFINED UsingMemory}
002869  {$I $$Shell(PInterfaces)Memory.p}
002870  {$ENDC}
002871  {$IFC UNDEFINED UsingPackages}
002872  {$I $$Shell(PInterfaces)Packages.p}
002873  {$ENDC}
002874  {$IFC UNDEFINED UsingAppleTalk}
002875  {$I $$Shell(PInterfaces)AppleTalk.p}
002876  {$ENDC}
002877  {$SETC UsingIncludes := CTBUtilitiesIncludes}
002878
002879  CONST
002880
002881  { version of Comm Toolbox Utilities }
002882  curCTBUVersion = 2;
002883
002884  {    Error codes/types    }
002885  ctbuGenericError = -1;
002886  ctbuNoErr = 0;
002887
002888  TYPE
002889  CTBUErr = OSErr;
002890
002891  CONST
002892
```

```
002893  { Choose return codes}
002894  chooseDisaster = -2;
002895  chooseFailed = -1;
002896  chooseAborted = 0;
002897  chooseOKMinor = 1;
002898  chooseOKMajor = 2;
002899  chooseCancel = 3;
002900
002901  { NuLookup return codes }
002902  nlOk = 0;
002903  nlCancel = 1;
002904  nlEject = 2;
002905
002906  { Name filter proc return codes }
002907  nameInclude = 1;
002908  nameDisable = 2;
002909  nameReject = 3;
002910
002911  { Zone filter proc return codes }
002912  zoneInclude = 1;
002913  zoneDisable = 2;
002914  zoneReject = 3;
002915
002916  {    Values for hookProc items        }
002917  hookOK = 1;
002918  hookCancel = 2;
002919  hookOutline = 3;
002920  hookTitle = 4;
002921  hookItemList = 5;
002922  hookZoneTitle = 6;
002923  hookZoneList = 7;
002924  hookLine = 8;
002925  hookVersion = 9;
002926  hookReserved1 = 10;
002927  hookReserved2 = 11;
002928  hookReserved3 = 12;
002929  hookReserved4 = 13;
002930
002931  {    "virtual" hookProc items    }
002932  hookNull = 100;
002933  hookItemRefresh = 101;
002934  hookZoneRefresh = 102;
002935  hookEject = 103;
002936  hookPreflight = 104;
002937  hookPostflight = 105;
002938  hookKeyBase = 1000;
002939
002940  TYPE
002941  {    NuLookup structures/constants    }
002942  NLTypeEntry = RECORD
002943   hIcon: Handle;
002944   typeStr: Str32;
002945   END;
002946
002947
002948  NLType = ARRAY [0..3] OF NLTypeEntry;
```

```
002949
002950  NBPReply = RECORD
002951   theEntity: EntityName;
002952   theAddr: AddrBlock;
002953   END;
002954
002955
002956  NameFilterProcPtr = ProcPtr;
002957  ZoneFilterProcPtr = ProcPtr;
002958
002959
002960  FUNCTION InitCTBUtilities: CTBUErr;
002961  FUNCTION CTBGetCTBVersion: INTEGER;
002962
002963
002964  FUNCTION StandardNBP(where: Point;prompt: Str255;numTypes: INTEGER;typeList: NLType;
002965   nameFilter: NameFilterProcPtr;zoneFilter: ZoneFilterProcPtr;hookProc: DlgHookProcPtr;
002966   VAR theReply: NBPReply): INTEGER;
002967
002968  FUNCTION CustomNBP(where: Point;prompt: Str255;numTypes: INTEGER;typeList: NLType;
002969   nameFilter: NameFilterProcPtr;zoneFilter: ZoneFilterProcPtr;hookProc: DlgHookProcPtr;
002970   userData: LONGINT;dialogID: INTEGER;filterProc: ModalFilterProcPtr;VAR theReply: NBPReply): INTEGER;
002971
002972
002973  {  Obsolete synonyms for above routines  }
002974  FUNCTION NuLookup(where: Point;prompt: Str255;numTypes: INTEGER;typeList: NLType;
002975   nameFilter: NameFilterProcPtr;zoneFilter: ZoneFilterProcPtr;hookProc: DlgHookProcPtr;
002976   VAR theReply: NBPReply): INTEGER;
002977
002978  FUNCTION NuPLookup(where: Point;prompt: Str255;numTypes: INTEGER;typeList: NLType;
002979   nameFilter: NameFilterProcPtr;zoneFilter: ZoneFilterProcPtr;hookProc: DlgHookProcPtr;
002980   userData: LONGINT;dialogID: INTEGER;filterProc: ModalFilterProcPtr;VAR theReply: NBPReply): INTEGER;
002981
002982
002983  {$ENDC} { UsingCTBUtilities }
002984
002985  {$IFC NOT UsingIncludes}
002986   END.
002987  {$ENDC}
002988
002989
002990  ### END OF FILE CTBUtilities.p
002991
```

```
002992
002993  ############################################################################
002994  ### FILE: CursorCtl.p
002995  ############################################################################
002996
002997  {
002998  Created: Tuesday, August 2, 1988 at 12:24 PM
002999      CursorCtl.p
003000      Pascal Interface to the Macintosh Libraries
003001
003002
003003      <<< CursorCtl - Cursor Control Interface File >>>
003004
003005
003006      Copyright Apple Computer, Inc. 1984-1988
003007      All rights reserved
003008
003009      This file contains:
003010
003011      InitCursorCtl(newCursors) - Init CursorCtl to load the 'acur' resource
003012      RotateCursor(counter) - Sequence through cursor frames for counter mod 32
003013      SpinCursor(increment) - Sequence mod 32 incrementing internal counter
003014      Hide_Cursor() - Hide the current cursor
003015      Show_Cursor(cursorKind) - Show the cursor
003016  }
003017
003018
003019  {$IFC UNDEFINED UsingIncludes}
003020  {$SETC UsingIncludes := 0}
003021  {$ENDC}
003022
003023  {$IFC NOT UsingIncludes}
003024      UNIT CursorCtl;
003025      INTERFACE
003026  {$ENDC}
003027
003028  {$IFC UNDEFINED UsingCursorCtl}
003029  {$SETC UsingCursorCtl := 1}
003030
003031
003032  TYPE
003033
003034  { Kinds of cursor supported by CursorCtl }
003035  Cursors = (HIDDEN_CURSOR,I_BEAM_CURSOR,CROSS_CURSOR,PLUS_CURSOR,WATCH_CURSOR,
003036      ARROW_CURSOR);
003037
003038  acurPtr = ^Acur;
003039  acurHandle = ^acurPtr;
003040  Acur = RECORD
003041      n: INTEGER;         {Number of cursors ("frames of film")}
003042      index: INTEGER;     { Next frame to show <for internal use>}
003043      frame1: INTEGER;    {'CURS' resource id for frame #1}
003044      fill1: INTEGER;     {<for internal use>}
003045      frame2: INTEGER;    {'CURS' resource id for frame #2}
003046      fill2: INTEGER;     {<for internal use>}
003047      frameN: INTEGER;    {'CURS' resource id for frame #N}
```

```
003048      fillN: INTEGER;     {<for internal use>}
003049      END;
003050
003051
003052
003053  PROCEDURE InitCursorCtl(newCursors: UNIV acurHandle);
003054  { Initialize the CursorCtl unit. This should be called once prior to calling
003055  RotateCursor or SpinCursor. It need not be called if only Hide_Cursor or
003056  Show_Cursor are used. If NewCursors is NULL, InitCursorCtl loads in the
003057  'acur' resource and the 'CURS' resources specified by the 'acur' resource
003058  ids.  If any of the resources cannot be loaded, the cursor will not be
003059  changed.
003060
003061  The 'acur' resource is assumed to either be in the currently running tool or
003062  application, or the MPW Shell for a tool, or in the System file.  The 'acur'
003063  resource id must be 0 for a tool or application, 1 for the Shell, and 2 for
003064  the System file.
003065
003066  If NewCursors is not NULL, it is ASSUMED to be a handle to an 'acur' formatted
003067  resource designated by the caller and it will be used instead of doing a
003068  GetResource on 'acur'. Note, if RotateCursor or SpinCursor are called without
003069  calling InitCursorCtl, then RotateCursor and SpinCursor will do the call for
003070  the user the first time it is called.  However, the possible disadvantage of
003071  using this technique is that the resource memory allocated may have
003072  undesirable affect (fragmentation?) on the application. Using InitCursorCtl
003073  has the advantage of causing the allocation at a specific time determined by
003074  the user.
003075
003076  Caution: InitCursorCtl MODIFIES the 'acur' resource in memory.  Specifically,
003077  it changes each FrameN/fillN integer pair to a handle to the corresponding
003078  'CURS' resource also in memory.  Thus if NewCursors is not NULL when
003079  InitCursorCtl is called, the caller must guarantee NewCursors always points to
003080  a "fresh" copy of an 'acur' resource.  This need only be of concern to a
003081  caller who wants to repeatly use multiple 'acur' resources during execution of
003082  their programs.
003083   }
003084
003085  PROCEDURE RotateCursor(counter: LONGINT);
003086  { RotateCursor is called to rotate the "I am active" "beach ball" cursor, or to
003087  animate whatever sequence of cursors set up by InitCursorCtl. The next cursor
003088  ("frame") is used when Counter % 32 = 0 (Counter is some kind of incrementing
003089  or decrementing index maintained by the caller). A positive counter sequences
003090  forward through the cursors (e.g., it rotates the "beach ball" cursor
003091  clockwise), and a negative cursor sequences through the cursors backwards
003092  (e.g., it rotates the "beach ball" cursor counterclockwise).  Note,
003093  RotateCursor just does a Mac SetCursor call for the proper cursor picture.
003094    It is assumed the cursor is visible from a prior Show_Cursor call.
003095   }
003096
003097  PROCEDURE SpinCursor(increment: INTEGER);
003098  { SpinCursor is similar in function to RotateCursor, except that instead of
003099  passing a counter, an Increment is passed an added to a counter maintained
003100  here.  SpinCursor is provided for those users who do not happen to have a
003101  convenient counter handy but still want to use the spinning "beach ball"
003102  cursor, or any sequence of cursors set up by InitCursorCtl.  A positive
003103  increment sequences forward through the curos (rotating the "beach ball"
```

```
003104  cursor clockwise), and a negative increment sequences backward through the
003105  cursors (rotating the "beach ball" cursor counter-clockwise).  A zero value
003106  for the increment resets the counter to zero.  Note, it is the increment, and
003107  not the value of the counter that determines the sequencing direction of the
003108  cursor (and hence the spin direction of the "beach ball" cursor).
003109   }
003110
003111  PROCEDURE Hide_Cursor;
003112  { Hide the cursor if it is showing.This is this unit's call to the Mac
003113  HideCursor routine.Thus the Mac cursor level is decremented by one when this
003114  routine is called.
003115   }
003116
003117  PROCEDURE Show_Cursor(cursorKind: Cursors);
003118  { Increment the cursor level, which may have been decremented by Hide_Cursor,
003119  and display the specified cursor if the level becomes 0 (it is never
003120  incremented beyond 0).The CursorKind is the kind of cursor to show.  It is
003121  one of the values HIDDEN_CURSOR, I_BEAM_CURSOR, CROSS_CURSOR, PLUS_CURSOR,
003122  WATCH_CURSOR, and ARROW_CURSOR. Except for HIDDEN_CURSOR, a Mac SetCursor is
003123  done for the specified cursor prior to doing a ShowCursor.  HIDDEN_CURSOR just
003124  causes a ShowCursor call.  Note, ARROW_CURSOR will only work correctly if
003125  there is already a grafPort set up pointed to by 0(A5).
003126   }
003127
003128
003129  {$ENDC}    { UsingCursorCtl }
003130
003131  {$IFC NOT UsingIncludes}
003132      END.
003133  {$ENDC}
003134
003135
003136  ### END OF FILE CursorCtl.p
003137
```

```
003138
003139  ############################################################################
003140  ### FILE: DatabaseAccess.p
003141  ############################################################################
003142
003143
003144  {
003145  Created: Tuesday, September 10, 1991 at 1:01 PM
003146   DatabaseAccess.p
003147   Pascal Interface to the Macintosh Libraries
003148
003149    Copyright Apple Computer, Inc. 1989-1991
003150    All rights reserved
003151  }
003152
003153
003154  {$IFC UNDEFINED UsingIncludes}
003155  {$SETC UsingIncludes := 0}
003156  {$ENDC}
003157
003158  {$IFC NOT UsingIncludes}
003159   UNIT DatabaseAccess;
003160   INTERFACE
003161  {$ENDC}
003162
003163  {$IFC UNDEFINED UsingDatabaseAccess}
003164  {$SETC UsingDatabaseAccess := 1}
003165
003166  {$I+}
003167  {$SETC DatabaseAccessIncludes := UsingIncludes}
003168  {$SETC UsingIncludes := 1}
003169  {$IFC UNDEFINED UsingResources}
003170  {$I $$Shell(PInterfaces)Resources.p}
003171  {$ENDC}
003172  {$SETC UsingIncludes := DatabaseAccessIncludes}
003173
003174  CONST
003175
003176  { error and status codes }
003177  rcDBNull = -800;
003178  rcDBValue = -801;
003179  rcDBError = -802;
003180  rcDBBadType = -803;
003181  rcDBBreak = -804;
003182  rcDBExec = -805;
003183  rcDBBadSessID = -806;
003184  rcDBBadSessNum = -807;          { bad session number for DBGetConnInfo }
003185  rcDBBadDDEV = -808;             { bad ddev specified on DBInit }
003186  rcDBAsyncNotSupp = -809;        { ddev does not support async calls }
003187  rcDBBadAsyncPB = -810;          { tried to kill a bad pb }
003188  rcDBNoHandler = -811;           { no app handler for specified data type }
003189  rcDBWrongVersion = -812;        { incompatible versions }
003190  rcDBPackNotInited = -813;       { attempt to call other routine before InitDBPack }
003191
003192  { messages for status functions for DBStartQuery }
003193  kDBUpdateWind = 0;
```

```
003194  kDBAboutToInit = 1;
003195  kDBInitComplete = 2;
003196  kDBSendComplete = 3;
003197  kDBExecComplete = 4;
003198  kDBStartQueryComplete = 5;
003199
003200  { messages for status functions for DBGetQueryResults }
003201  kDBGetItemComplete = 6;
003202  kDBGetQueryResultsComplete = 7;
003203
003204  { data type codes }
003205  typeNone = 'none';
003206  typeDate = 'date';
003207  typeTime = 'time';
003208  typeTimeStamp = 'tims';
003209  typeDecimal = 'deci';
003210  typeMoney = 'mone';
003211  typeVChar = 'vcha';
003212  typeVBin = 'vbin';
003213  typeLChar = 'lcha';
003214  typeLBin = 'lbin';
003215  typeDiscard = 'disc';
003216
003217  { "dummy" types for DBResultsToText }
003218  typeUnknown = 'unkn';
003219  typeColBreak = 'colb';
003220  typeRowBreak = 'rowb';
003221
003222  { pass this in to DBGetItem for any data type }
003223  typeAnyType = 0;
003224
003225  { infinite timeout value for DBGetItem }
003226  kDBWaitForever = -1;
003227
003228  {  flags for DBGetItem  }
003229  kDBLastColFlag = $0001;
003230  kDBNullFlag = $0004;
003231
003232  TYPE
003233  DBType = OSType;
003234
003235  { structure for asynchronous parameter block }
003236  DBAsyncParmBlkPtr = ^DBAsyncParamBlockRec;
003237  DBAsyncParamBlockRec = RECORD
003238   completionProc: ProcPtr;       { pointer to completion routine }
003239   result: OSErr;                 { result of call }
003240   userRef: LONGINT;              { for application's use }
003241   ddevRef: LONGINT;              { for ddev's use }
003242   reserved: LONGINT;             { for internal use }
003243   END;
003244
003245  { structure for resource list in QueryRecord }
003246  ResListElem = RECORD
003247   theType: ResType;              { resource type }
003248   id: INTEGER;                   { resource id }
003249   END;
```

```
003250
003251  ResListPtr = ^ResListArray;
003252  ResListHandle = ^ResListPtr;
003253
003254  ResListArray = ARRAY [0..255] OF ResListElem;
003255
003256  { structure for query list in QueryRecord }
003257  QueryListPtr = ^QueryArray;
003258  QueryListHandle = ^QueryListPtr;
003259
003260  QueryArray = ARRAY [0..255] OF Handle;
003261
003262  QueryPtr = ^QueryRecord;
003263  QueryHandle = ^QueryPtr;
003264  QueryRecord = RECORD
003265   version: INTEGER;              { version }
003266   id: INTEGER;                   { id of 'qrsc' this came from }
003267   queryProc: Handle;             { handle to query def proc }
003268   ddevName: Str63;               { ddev name }
003269   host: Str255;                  { host name }
003270   user: Str255;                  { user name }
003271   password: Str255;              { password }
003272   connStr: Str255;               { connection string }
003273   currQuery: INTEGER;            { index of current query }
003274   numQueries: INTEGER;           { number of queries in list }
003275   queryList: QueryListHandle;    { handle to array of handles to text }
003276   numRes: INTEGER;               { number of resources in list }
003277   resList: ResListHandle;        { handle to array of resource list elements }
003278   dataHandle: Handle;            { for use by query def proc }
003279   refCon: LONGINT;               { for use by application }
003280   END;
003281
003282  { structure of column types array in ResultsRecord }
003283  ColTypesPtr = ^ColTypesArray;
003284  ColTypesHandle = ^ColTypesPtr;
003285
003286  ColTypesArray = ARRAY [0..255] OF DBType;
003287
003288  { structure for column info in ResultsRecord }
003289  DBColInfoRecord = RECORD
003290   len: INTEGER;
003291   places: INTEGER;
003292   flags: INTEGER;
003293   END;
003294
003295  ColInfoPtr = ^ColInfoArray;
003296  ColInfoHandle = ^ColInfoPtr;
003297
003298  ColInfoArray = ARRAY [0..255] OF DBColInfoRecord;
003299
003300  { structure of results returned by DBGetResults }
003301  ResultsRecord = RECORD
003302   numRows: INTEGER;              { number of rows in result }
003303   numCols: INTEGER;              { number of columns per row }
003304   colTypes: ColTypesHandle;      { data type array }
003305   colData: Handle;               { actual results }
```

```
003306   colInfo: ColInfoHandle;        { DBColInfoRecord array }
003307   END;
003308
003309
003310  FUNCTION InitDBPack: OSErr;
003311   INLINE $3F3C,$0004,$303C,$0100,$A82F;
003312  FUNCTION DBInit(VAR sessID: LONGINT;ddevName: Str63;host: Str255;user: Str255;
003313   passwd: Str255;connStr: Str255;asyncPB: DBAsyncParmBlkPtr): OSErr;
003314   INLINE $303C,$0E02,$A82F;
003315  FUNCTION DBEnd(sessID: LONGINT;asyncPB: DBAsyncParmBlkPtr): OSErr;
003316   INLINE $303C,$0403,$A82F;
003317  FUNCTION DBGetConnInfo(sessID: LONGINT;sessNum: INTEGER;VAR returnedID: LONGINT;
003318   VAR version: LONGINT;VAR ddevName: Str63;VAR host: Str255;VAR user: Str255;
003319   VAR network: Str255;VAR connStr: Str255;VAR start: LONGINT;VAR state: OSErr;
003320   asyncPB: DBAsyncParmBlkPtr): OSErr;
003321   INLINE $303C,$1704,$A82F;
003322  FUNCTION DBGetSessionNum(sessID: LONGINT;VAR sessNum: INTEGER;asyncPB: DBAsyncParmBlkPtr): OSErr;
003323   INLINE $303C,$0605,$A82F;
003324  FUNCTION DBSend(sessID: LONGINT;text: Ptr;len: INTEGER;asyncPB: DBAsyncParmBlkPtr): OSErr;
003325   INLINE $303C,$0706,$A82F;
003326  FUNCTION DBSendItem(sessID: LONGINT;dataType: DBType;len: INTEGER;places: INTEGER;
003327   flags: INTEGER;buffer: Ptr;asyncPB: DBAsyncParmBlkPtr): OSErr;
003328   INLINE $303C,$0B07,$A82F;
003329  FUNCTION DBExec(sessID: LONGINT;asyncPB: DBAsyncParmBlkPtr): OSErr;
003330   INLINE $303C,$0408,$A82F;
003331  FUNCTION DBState(sessID: LONGINT;asyncPB: DBAsyncParmBlkPtr): OSErr;
003332   INLINE $303C,$0409,$A82F;
003333  FUNCTION DBGetErr(sessID: LONGINT;VAR err1: LONGINT;VAR err2: LONGINT;VAR item1: Str255;
003334   VAR item2: Str255;VAR errorMsg: Str255;asyncPB: DBAsyncParmBlkPtr): OSErr;
003335   INLINE $303C,$0E0A,$A82F;
003336  FUNCTION DBBreak(sessID: LONGINT;abort: BOOLEAN;asyncPB: DBAsyncParmBlkPtr): OSErr;
003337   INLINE $303C,$050B,$A82F;
003338  FUNCTION DBGetItem(sessID: LONGINT;timeout: LONGINT;VAR dataType: DBType;
003339   VAR len: INTEGER;VAR places: INTEGER;VAR flags: INTEGER;buffer: Ptr;asyncPB: DBAsyncParmBlkPtr): OSErr;
003340   INLINE $303C,$100C,$A82F;
003341  FUNCTION DBUnGetItem(sessID: LONGINT;asyncPB: DBAsyncParmBlkPtr): OSErr;
003342   INLINE $303C,$040D,$A82F;
003343  FUNCTION DBKill(asyncPB: DBAsyncParmBlkPtr): OSErr;
003344   INLINE $303C,$020E,$A82F;
003345  FUNCTION DBGetNewQuery(queryID: INTEGER;VAR query: QueryHandle): OSErr;
003346   INLINE $303C,$030F,$A82F;
003347  FUNCTION DBDisposeQuery(query: QueryHandle): OSErr;
003348   INLINE $303C,$0210,$A82F;
003349  FUNCTION DBStartQuery(VAR sessID: LONGINT;query: QueryHandle;statusProc: ProcPtr;
003350   asyncPB: DBAsyncParmBlkPtr): OSErr;
003351   INLINE $303C,$0811,$A82F;
003352  FUNCTION DBGetQueryResults(sessID: LONGINT;VAR results: ResultsRecord;timeout: LONGINT;
003353   statusProc: ProcPtr;asyncPB: DBAsyncParmBlkPtr): OSErr;
003354   INLINE $303C,$0A12,$A82F;
003355  FUNCTION DBResultsToText(results: ResultsRecord;VAR theText: Handle): OSErr;
003356   INLINE $303C,$0413,$A82F;
003357  FUNCTION DBInstallResultHandler(dataType: DBType;theHandler: ProcPtr;isSysHandler: BOOLEAN): OSErr;
003358   INLINE $303C,$0514,$A82F;
003359  FUNCTION DBRemoveResultHandler(dataType: DBType): OSErr;
003360   INLINE $303C,$0215,$A82F;
003361  FUNCTION DBGetResultHandler(dataType: DBType;VAR theHandler: ProcPtr;getSysHandler: BOOLEAN): OSErr;
```

```
003362   INLINE $303C,$0516,$A82F;
003363
003364
003365  {$ENDC} { UsingDatabaseAccess }
003366
003367  {$IFC NOT UsingIncludes}
003368   END.
003369  {$ENDC}
003370
003371
003372  ### END OF FILE DatabaseAccess.p
003373
```

```
003374
003375  ##########################################################################
003376  ### FILE: Desk.p
003377  ##########################################################################
003378
003379  {
003380  Created: Sunday, January 6, 1991 at 10:27 PM
003381      Desk.p
003382      Pascal Interface to the Macintosh Libraries
003383
003384          Copyright Apple Computer, Inc.    1985-1989
003385          All rights reserved
003386  }
003387
003388
003389  {$IFC UNDEFINED UsingIncludes}
003390  {$SETC UsingIncludes := 0}
003391  {$ENDC}
003392
003393  {$IFC NOT UsingIncludes}
003394      UNIT Desk;
003395      INTERFACE
003396  {$ENDC}
003397
003398  {$IFC UNDEFINED UsingDesk}
003399  {$SETC UsingDesk := 1}
003400
003401  {$I+}
003402  {$SETC DeskIncludes := UsingIncludes}
003403  {$SETC UsingIncludes := 1}
003404  {$IFC UNDEFINED UsingTypes}
003405  {$I $$Shell(PInterfaces)Types.p}
003406  {$ENDC}
003407  {$IFC UNDEFINED UsingQuickdraw}
003408  {$I $$Shell(PInterfaces)Quickdraw.p}
003409  {$ENDC}
003410  {$IFC UNDEFINED UsingEvents}
003411  {$I $$Shell(PInterfaces)Events.p}
003412  {$ENDC}
003413  {$SETC UsingIncludes := DeskIncludes}
003414
003415  CONST
003416  accEvent = 64;
003417  accRun = 65;
003418  accCursor = 66;
003419  accMenu = 67;
003420  accUndo = 68;
003421  accCut = 70;
003422  accCopy = 71;
003423  accPaste = 72;
003424  accClear = 73;
003425  goodbye = -1;   {goodbye message}
003426
003427  FUNCTION OpenDeskAcc(deskAccName: Str255): INTEGER;
003428      INLINE $A9B6;
003429  PROCEDURE CloseDeskAcc(refNum: INTEGER);
```

```
003430      INLINE $A9B7;
003431  PROCEDURE SystemClick(theEvent: EventRecord;theWindow: WindowPtr);
003432      INLINE $A9B3;
003433  FUNCTION SystemEdit(editCmd: INTEGER): BOOLEAN;
003434      INLINE $A9C2;
003435  PROCEDURE SystemTask;
003436      INLINE $A9B4;
003437  FUNCTION SystemEvent(theEvent: EventRecord): BOOLEAN;
003438      INLINE $A9B2;
003439  PROCEDURE SystemMenu(menuResult: LONGINT);
003440      INLINE $A9B5;
003441
003442
003443  {$ENDC}    { UsingDesk }
003444
003445  {$IFC NOT UsingIncludes}
003446      END.
003447  {$ENDC}
003448
003449
003450  ### END OF FILE Desk.p
003451
```

```
003452
003453  ############################################################################
003454  ### FILE: DeskBus.p
003455  ############################################################################
003456
003457
003458  {
003459  Created: Sunday, September 15, 1991 at 10:14 PM
003460   DeskBus.p
003461   Pascal Interface to the Macintosh Libraries
003462
003463    Copyright Apple Computer, Inc. 1987-1991
003464    All rights reserved
003465  }
003466
003467
003468  {$IFC UNDEFINED UsingIncludes}
003469  {$SETC UsingIncludes := 0}
003470  {$ENDC}
003471
003472  {$IFC NOT UsingIncludes}
003473   UNIT DeskBus;
003474   INTERFACE
003475  {$ENDC}
003476
003477  {$IFC UNDEFINED UsingDeskBus}
003478  {$SETC UsingDeskBus := 1}
003479
003480  {$I+}
003481  {$SETC DeskBusIncludes := UsingIncludes}
003482  {$SETC UsingIncludes := 1}
003483  {$IFC UNDEFINED UsingTypes}
003484  {$I $$Shell(PInterfaces)Types.p}
003485  {$ENDC}
003486  {$SETC UsingIncludes := DeskBusIncludes}
003487
003488  TYPE
003489  ADBAddress = SignedByte;
003490
003491  ADBOpBPtr = ^ADBOpBlock;
003492  ADBOpBlock = RECORD
003493   dataBuffPtr: Ptr;          {address of data buffer}
003494   opServiceRtPtr: Ptr;       {service routine pointer}
003495   opDataAreaPtr: Ptr;        {optional data area address}
003496   END;
003497
003498  ADBDBlkPtr = ^ADBDataBlock;
003499  ADBDataBlock = PACKED RECORD
003500   devType: SignedByte;       {device type}
003501   origADBAddr: SignedByte;   {original ADB Address}
003502   dbServiceRtPtr: Ptr;       {service routine pointer}
003503   dbDataAreaAddr: Ptr;       {data area address}
003504   END;
003505
003506  ADBSInfoPtr = ^ADBSetInfoBlock;
003507  ADBSetInfoBlock = RECORD
```

```
003508   siServiceRtPtr: Ptr;       {service routine pointer}
003509   siDataAreaAddr: Ptr;       {data area address}
003510   END;
003511
003512
003513  PROCEDURE ADBReInit;
003514   INLINE $A07B;
003515  FUNCTION ADBOp(data: Ptr;compRout: ProcPtr;buffer: Ptr;commandNum: INTEGER): OSErr;
003516  FUNCTION CountADBs: INTEGER;
003517   INLINE $A077,$3E80;
003518  FUNCTION GetIndADB(VAR info: ADBDataBlock;devTableIndex: INTEGER): ADBAddress;
003519  FUNCTION GetADBInfo(VAR info: ADBDataBlock;adbAddr: ADBAddress): OSErr;
003520  FUNCTION SetADBInfo(VAR info: ADBSetInfoBlock;adbAddr: ADBAddress): OSErr;
003521
003522
003523  {$ENDC} { UsingDeskBus }
003524
003525  {$IFC NOT UsingIncludes}
003526   END.
003527  {$ENDC}
003528
003529
003530  ### END OF FILE DeskBus.p
003531
```

```
003532
003533  ##########################################################################
003534  ### FILE: Devices.p
003535  ##########################################################################
003536
003537  {
003538  Created: Sunday, January 6, 1991 at 10:28 PM
003539      Devices.p
003540      Pascal Interface to the Macintosh Libraries
003541
003542          Copyright Apple Computer, Inc.     1985-1990
003543          All rights reserved
003544  }
003545
003546
003547  {$IFC UNDEFINED UsingIncludes}
003548  {$SETC UsingIncludes := 0}
003549  {$ENDC}
003550
003551  {$IFC NOT UsingIncludes}
003552      UNIT Devices;
003553      INTERFACE
003554  {$ENDC}
003555
003556  {$IFC UNDEFINED UsingDevices}
003557  {$SETC UsingDevices := 1}
003558
003559  {$I+}
003560  {$SETC DevicesIncludes := UsingIncludes}
003561  {$SETC UsingIncludes := 1}
003562  {$IFC UNDEFINED UsingOSUtils}
003563  {$I $$Shell(PInterfaces)OSUtils.p}
003564  {$ENDC}
003565  {$IFC UNDEFINED UsingFiles}
003566  {$I $$Shell(PInterfaces)Files.p}
003567  {$ENDC}
003568  {$IFC UNDEFINED UsingQuickdraw}
003569  {$I $$Shell(PInterfaces)Quickdraw.p}
003570  {$ENDC}
003571  {$SETC UsingIncludes := DevicesIncludes}
003572
003573  CONST
003574  newSelMsg = 12;
003575  fillListMsg = 13;
003576  getSelMsg = 14;
003577  selectMsg = 15;
003578  deselectMsg = 16;
003579  terminateMsg = 17;
003580  buttonMsg = 19;
003581  chooserID = 1;
003582  initDev = 0;        {Time for cdev to initialize itself}
003583  hitDev = 1;         {Hit on one of my items}
003584  closeDev = 2;       {Close yourself}
003585  nulDev = 3;         {Null event}
003586  updateDev = 4;      {Update event}
003587  activDev = 5;       {Activate event}
```

```
003588  deactivDev = 6;     {Deactivate event}
003589  keyEvtDev = 7;      {Key down/auto key}
003590  macDev = 8;         {Decide whether or not to show up}
003591  undoDev = 9;
003592  cutDev = 10;
003593  copyDev = 11;
003594  pasteDev = 12;
003595  clearDev = 13;
003596  cursorDev = 14;
003597  cdevGenErr = -1;    {General error; gray cdev w/o alert}
003598  cdevMemErr = 0;     {Memory shortfall; alert user please}
003599  cdevResErr = 1;     {Couldn't get a needed resource; alert}
003600  cdevUnset = 3;      { cdevValue is initialized to this}
003601
003602  { Monitors control panel messages }
003603  initMsg = 1;        {initialization}
003604  okMsg = 2;          {user clicked OK button}
003605  cancelMsg = 3;      {user clicked Cancel button}
003606  hitMsg = 4;         {user clicked control in Options dialog}
003607  nulMsg = 5;         {periodic event}
003608  updateMsg = 6;      {update event}
003609  activateMsg = 7;    {not used}
003610  deactivateMsg = 8;  {not used}
003611  keyEvtMsg = 9;      {keyboard event}
003612  superMsg = 10;      {show superuser controls}
003613  normalMsg = 11;     {show only normal controls}
003614  startupMsg = 12;    {code has been loaded}
003615
003616  TYPE
003617  DCtlPtr = ^DCtlEntry;
003618  DCtlHandle = ^DCtlPtr;
003619  DCtlEntry = RECORD
003620      dCtlDriver: Ptr;
003621      dCtlFlags: INTEGER;
003622      dCtlQHdr: QHdr;
003623      dCtlPosition: LONGINT;
003624      dCtlStorage: Handle;
003625      dCtlRefNum: INTEGER;
003626      dCtlCurTicks: LONGINT;
003627      dCtlWindow: WindowPtr;
003628      dCtlDelay: INTEGER;
003629      dCtlEMask: INTEGER;
003630      dCtlMenu: INTEGER;
003631      END;
003632
003633  AuxDCEPtr = ^AuxDCE;
003634  AuxDCEHandle = ^AuxDCEPtr;
003635  AuxDCE = PACKED RECORD
003636      dCtlDriver: Ptr;
003637      dCtlFlags: INTEGER;
003638      dCtlQHdr: QHdr;
003639      dCtlPosition: LONGINT;
003640      dCtlStorage: Handle;
003641      dCtlRefNum: INTEGER;
003642      dCtlCurTicks: LONGINT;
003643      dCtlWindow: GrafPtr;
```

```
003644      dCtlDelay: INTEGER;
003645      dCtlEMask: INTEGER;
003646      dCtlMenu: INTEGER;
003647      dCtlSlot: Byte;
003648      dCtlSlotId: Byte;
003649      dCtlDevBase: LONGINT;
003650      dCtlOwner: Ptr;
003651      dCtlExtDev: Byte;
003652      fillByte: Byte;
003653      END;
003654
003655
003656  FUNCTION GetDCtlEntry(refNum: INTEGER): DCtlHandle;
003657  FUNCTION SetChooserAlert(f: BOOLEAN): BOOLEAN;
003658  FUNCTION OpenDriver(name: Str255;VAR drvrRefNum: INTEGER): OSErr;
003659  FUNCTION CloseDriver(refNum: INTEGER): OSErr;
003660  FUNCTION Control(refNum: INTEGER;csCode: INTEGER;csParamPtr: Ptr): OSErr;
003661  FUNCTION Status(refNum: INTEGER;csCode: INTEGER;csParamPtr: Ptr): OSErr;
003662  FUNCTION KillIO(refNum: INTEGER): OSErr;
003663  FUNCTION PBControl(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
003664  FUNCTION PBControlSync(paramBlock: ParmBlkPtr): OSErr;
003665      INLINE $205F,$A004,$3E80;
003666  FUNCTION PBControlAsync(paramBlock: ParmBlkPtr): OSErr;
003667      INLINE $205F,$A404,$3E80;
003668  FUNCTION PBStatus(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
003669  FUNCTION PBStatusSync(paramBlock: ParmBlkPtr): OSErr;
003670      INLINE $205F,$A005,$3E80;
003671  FUNCTION PBStatusAsync(paramBlock: ParmBlkPtr): OSErr;
003672      INLINE $205F,$A405,$3E80;
003673  FUNCTION PBKillIO(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
003674  FUNCTION PBKillIOSync(paramBlock: ParmBlkPtr): OSErr;
003675      INLINE $205F,$A006,$3E80;
003676  FUNCTION PBKillIOAsync(paramBlock: ParmBlkPtr): OSErr;
003677      INLINE $205F,$A406,$3E80;
003678
003679
003680  {$ENDC}    { UsingDevices }
003681
003682  {$IFC NOT UsingIncludes}
003683      END.
003684  {$ENDC}
003685
003686
003687  ### END OF FILE Devices.p
003688
```

```
003689
003690  ############################################################################
003691  ### FILE: Dialogs.p
003692  ############################################################################
003693
003694
003695  {
003696  Created: Thursday, September 12, 1991 at 2:47 PM
003697   Dialogs.p
003698   Pascal Interface to the Macintosh Libraries
003699
003700    Copyright Apple Computer, Inc.  1985-1991
003701    All rights reserved
003702  }
003703
003704
003705  {$IFC UNDEFINED UsingIncludes}
003706  {$SETC UsingIncludes := 0}
003707  {$ENDC}
003708
003709  {$IFC NOT UsingIncludes}
003710   UNIT Dialogs;
003711   INTERFACE
003712  {$ENDC}
003713
003714  {$IFC UNDEFINED UsingDialogs}
003715  {$SETC UsingDialogs := 1}
003716
003717  {$I+}
003718  {$SETC DialogsIncludes := UsingIncludes}
003719  {$SETC UsingIncludes := 1}
003720  {$IFC UNDEFINED UsingWindows}
003721  {$I $$Shell(PInterfaces)Windows.p}
003722  {$ENDC}
003723  {$IFC UNDEFINED UsingTextEdit}
003724  {$I $$Shell(PInterfaces)TextEdit.p}
003725  {$ENDC}
003726  {$SETC UsingIncludes := DialogsIncludes}
003727
003728  CONST
003729  ctrlItem = 4;
003730  btnCtrl = 0;
003731  chkCtrl = 1;
003732  radCtrl = 2;
003733  resCtrl = 3;
003734  statText = 8;
003735  editText = 16;
003736  iconItem = 32;
003737  picItem = 64;
003738  userItem = 0;
003739  itemDisable = 128;
003740  ok = 1;
003741  cancel = 2;
003742  stopIcon = 0;
003743  noteIcon = 1;
003744  cautionIcon = 2;
```

```
003745
003746  TYPE
003747  { Dialog Item List Manipulation Constants }
003748  DITLMethod = INTEGER;
003749
003750  CONST
003751  overlayDITL = 0;
003752  appendDITLRight = 1;
003753  appendDITLBottom = 2;
003754
003755  TYPE
003756  StageList = PACKED RECORD
003757   boldItm4: 0..1;                {default button item number - 1}
003758   boxDrwn4: BOOLEAN;             {true if alert box to be drawn}
003759   sound4: 0..3;                  {sound number}
003760   boldItm3: 0..1;
003761   boxDrwn3: BOOLEAN;
003762   sound3: 0..3;
003763   boldItm2: 0..1;
003764   boxDrwn2: BOOLEAN;
003765   sound2: 0..3;
003766   boldItm1: 0..1;
003767   boxDrwn1: BOOLEAN;
003768   sound1: 0..3;
003769   END;
003770
003771
003772  DialogPtr = WindowPtr;
003773  ResumeProcPtr = ProcPtr;        { PROCEDURE Resume; }
003774  SoundProcPtr = ProcPtr;         { PROCEDURE DoSound; }
003775  ModalFilterProcPtr = ProcPtr;   { FUNCTION Filter(theDialog: DialogPtr; VAR theEvent: EventRecord; VAR itemHit: INTEGER): BOOLEAN; }
003776
003777  DialogPeek = ^DialogRecord;
003778  DialogRecord = RECORD
003779   window: WindowRecord;
003780   items: Handle;
003781   textH: TEHandle;
003782   editField: INTEGER;
003783   editOpen: INTEGER;
003784   aDefItem: INTEGER;
003785   END;
003786
003787  DialogTPtr = ^DialogTemplate;
003788  DialogTHndl = ^DialogTPtr;
003789  DialogTemplate = RECORD
003790   boundsRect: Rect;
003791   procID: INTEGER;
003792   visible: BOOLEAN;
003793   filler1: BOOLEAN;
003794   goAwayFlag: BOOLEAN;
003795   filler2: BOOLEAN;
003796   refCon: LONGINT;
003797   itemsID: INTEGER;
003798   title: Str255;
003799   END;
003800
```

```
003801  AlertTPtr = ^AlertTemplate;
003802  AlertTHndl = ^AlertTPtr;
003803  AlertTemplate = RECORD
003804   boundsRect: Rect;
003805   itemsID: INTEGER;
003806   stages: StageList;
003807   END;
003808
003809
003810  PROCEDURE InitDialogs(resumeProc: ResumeProcPtr);
003811   INLINE $A97B;
003812  PROCEDURE ErrorSound(soundProc: SoundProcPtr);
003813   INLINE $A98C;
003814  FUNCTION NewDialog(wStorage: Ptr;boundsRect: Rect;title: Str255;visible: BOOLEAN;
003815   procID: INTEGER;behind: WindowPtr;goAwayFlag: BOOLEAN;refCon: LONGINT;
003816   itmLstHndl: Handle): DialogPtr;
003817   INLINE $A97D;
003818  FUNCTION GetNewDialog(dialogID: INTEGER;dStorage: Ptr;behind: WindowPtr): DialogPtr;
003819   INLINE $A97C;
003820  PROCEDURE CloseDialog(theDialog: DialogPtr);
003821   INLINE $A982;
003822  PROCEDURE DisposDialog(theDialog: DialogPtr);
003823   INLINE $A983;
003824  PROCEDURE DisposeDialog(theDialog: DialogPtr);
003825   INLINE $A983;
003826  PROCEDURE CouldDialog(dialogID: INTEGER);
003827   INLINE $A979;
003828  PROCEDURE FreeDialog(dialogID: INTEGER);
003829   INLINE $A97A;
003830  PROCEDURE ParamText(param0: Str255;param1: Str255;param2: Str255;param3: Str255);
003831   INLINE $A98B;
003832  PROCEDURE ModalDialog(filterProc: ModalFilterProcPtr;VAR itemHit: INTEGER);
003833   INLINE $A991;
003834  FUNCTION IsDialogEvent(theEvent: EventRecord): BOOLEAN;
003835   INLINE $A97F;
003836  FUNCTION DialogSelect(theEvent: EventRecord;VAR theDialog: DialogPtr;VAR itemHit: INTEGER): BOOLEAN;
003837   INLINE $A980;
003838  PROCEDURE DrawDialog(theDialog: DialogPtr);
003839   INLINE $A981;
003840  PROCEDURE UpdtDialog(theDialog: DialogPtr;updateRgn: RgnHandle);
003841   INLINE $A978;
003842  PROCEDURE UpdateDialog(theDialog: DialogPtr;updateRgn: RgnHandle);
003843   INLINE $A978;
003844  FUNCTION Alert(alertID: INTEGER;filterProc: ModalFilterProcPtr): INTEGER;
003845   INLINE $A985;
003846  FUNCTION StopAlert(alertID: INTEGER;filterProc: ModalFilterProcPtr): INTEGER;
003847   INLINE $A986;
003848  FUNCTION NoteAlert(alertID: INTEGER;filterProc: ModalFilterProcPtr): INTEGER;
003849   INLINE $A987;
003850  FUNCTION CautionAlert(alertID: INTEGER;filterProc: ModalFilterProcPtr): INTEGER;
003851   INLINE $A988;
003852  PROCEDURE CouldAlert(alertID: INTEGER);
003853   INLINE $A989;
003854  PROCEDURE FreeAlert(alertID: INTEGER);
003855   INLINE $A98A;
003856  PROCEDURE GetDItem(theDialog: DialogPtr;itemNo: INTEGER;VAR itemType: INTEGER;
```

```
003857   VAR item: Handle;VAR box: Rect);
003858   INLINE $A98D;
003859  PROCEDURE SetDItem(theDialog: DialogPtr;itemNo: INTEGER;itemType: INTEGER;
003860   item: Handle;box: Rect);
003861   INLINE $A98E;
003862  PROCEDURE HideDItem(theDialog: DialogPtr;itemNo: INTEGER);
003863   INLINE $A827;
003864  PROCEDURE ShowDItem(theDialog: DialogPtr;itemNo: INTEGER);
003865   INLINE $A828;
003866  PROCEDURE SelIText(theDialog: DialogPtr;itemNo: INTEGER;strtSel: INTEGER;
003867   endSel: INTEGER);
003868   INLINE $A97E;
003869  PROCEDURE GetIText(item: Handle;VAR text: Str255);
003870   INLINE $A990;
003871  PROCEDURE SetIText(item: Handle;text: Str255);
003872   INLINE $A98F;
003873  FUNCTION FindDItem(theDialog: DialogPtr;thePt: Point): INTEGER;
003874   INLINE $A984;
003875  FUNCTION NewCDialog(dStorage: Ptr;boundsRect: Rect;title: Str255;visible: BOOLEAN;
003876   procID: INTEGER;behind: WindowPtr;goAwayFlag: BOOLEAN;refCon: LONGINT;
003877   items: Handle): DialogPtr;
003878   INLINE $AA4B;
003879  FUNCTION GetAlrtStage: INTEGER;
003880   INLINE $3EB8,$0A9A;
003881  PROCEDURE ResetAlrtStage;
003882   INLINE $4278,$0A9A;
003883  PROCEDURE DlgCut(theDialog: DialogPtr);
003884  PROCEDURE DlgPaste(theDialog: DialogPtr);
003885  PROCEDURE DlgCopy(theDialog: DialogPtr);
003886  PROCEDURE DlgDelete(theDialog: DialogPtr);
003887  PROCEDURE SetDAFont(fontNum: INTEGER);
003888   INLINE $31DF,$0AFA;
003889
003890  PROCEDURE AppendDITL(theDialog: DialogPtr;theHandle: Handle;method: DITLMethod);
003891  FUNCTION CountDITL(theDialog: DialogPtr): INTEGER;
003892  PROCEDURE ShortenDITL(theDialog: DialogPtr;numberItems: INTEGER);
003893
003894
003895  {$ENDC} { UsingDialogs }
003896
003897  {$IFC NOT UsingIncludes}
003898   END.
003899  {$ENDC}
003900
003901
003902  ### END OF FILE Dialogs.p
003903
```

```
003904
003905  ############################################################################
003906  ### FILE: DisAsmLookup.p
003907  ############################################################################
003908
003909  {
003910      Created: Wednesday, November 1, 1989
003911      DisAsmLookup.p
003912      Pascal Interface to the Macintosh Libraries
003913
003914      Copyright Apple Computer, Inc.  1987-1990
003915      All rights reserved
003916  }
003917
003918  {$IFC UNDEFINED UsingIncludes}
003919  {$SETC UsingIncludes := 0}
003920  {$ENDC}
003921
003922  {$IFC NOT UsingIncludes}
003923      UNIT DisAsmLookup;
003924      INTERFACE
003925  {$ENDC}
003926
003927  {$IFC UNDEFINED UsingDisAsmLookup}
003928  {$SETC UsingDisAsmLookup := 1}
003929
003930  {$I+}
003931  {$SETC DisAsmLookupIncludes := UsingIncludes}
003932  {$SETC UsingIncludes := 1}
003933  {$IFC UNDEFINED UsingTypes}
003934  {$I $$Shell(PInterfaces)Types.p}
003935  {$ENDC}
003936  {$SETC UsingIncludes := DisAsmLookupIncludes}
003937
003938
003939  TYPE
003940      LookupRegs  = (_A0_, _A1_, _A2_, _A3_, _A4_, _A5_, _A6_, _A7_,
003941                                  _PC_, _ABS_, _TRAP_, _IMM_);
003942
003943  (*-----------------------------------------------------------------------*)
003944
003945  PROCEDURE Disassembler(      DstAdjust:  LongInt;               {addr correction}
003946                                           VAR BytesUsed:  Integer;               {bytes used up  }
003947                                                   FirstByte:  UNIV Ptr;              {starting byte  }
003948                                           VAR Opcode:         UNIV Str255;       {mnemonic       }
003949                                           VAR Operand:    UNIV Str255;      {operand        }
003950                                           VAR Comment:    UNIV Str255;        {comment        }
003951                                                   LookupProc: UNIV Ptr);             {search proc    }
003952      (*
003953       Disassembler is a Pascal routine to be called to disassemble a sequence
003954       of bytes.  All MC68xxx, MC68881, and MC68851 instructions are supported.
003955       The sequence of bytes to be disassembled are pointed to by FirstByte.
003956       BytesUsed bytes starting at FirstByte are consumed by the disassembly,
003957       and the Opcode, Operand, and Comment strings returned as NULL TERMINATED
003958       Pascal strings (for easier manipulation with C). The caller is then free
003959       to format or use the output strings any way appropriate to the
```

```
003960      application.
003961
003962      Depending on the opcode and effective address(s) (EA's) to be
003963      disassembled, the Opcode, Operand, and Comment strings contain the
003964      following information:
003965
003966      Case                    Opcode    Operand    Comment
003967      ========================================================================
003968      Non PC-relative EA's    op.sz     EA's            ; 'c…' (for immediates)
003969      PC-relative EA's        op.sz     EA's       ; address
003970      Toolbox traps           DC.W      $AXXX      ; TB XXXX
003971      OS traps                DC.W      $AXXX      ; OS XXXX
003972      Invalid bytes           DC.W      $XXXX      ; ????
003973      ========================================================================
003974
003975      For valid disassembly of processor instructions the appropriate MC68XXX
003976      opcode mnemonic is generated for the Opcode string along with a size
003977      attribute when required.  The source and destination EA's are generated
003978      as the Operand along with a possible comment. Comments start with a ';'.
003979      Traps use a DC.W assembler directive as the Opcode with the trap word
003980      as the Operand and a comment indicating whether the trap is a toolbox or
003981      OS trap and what the trap number is.  As described later the caller can
003982      generate symbolic substitutions into EA's and provide names for traps.
003983
003984      Invalid instructions cause the string 'DC.W' to be returned in the
003985      Opcode string. Operand is '$XXXX' (the invalid word) with a comment of
003986      '; ????'.  BytesUsed is 2.  This is similar to the trap call case except
003987      for the comment.
003988
003989      Note, the Operand EA's is syntatically similar to but NOT COMPATIBLE
003990      with the MPW assembler!    This is because the Disassembler generates
003991      byte hex constants as "$XX" and word hex constants as "$XXXX".  Negative
003992      values (e.g., $FF or $FFFF) produced by the Disassembler are treated as
003993      long word values by the MPW assembler.  Thus it is assumed that
003994      Disassembler output will NOT be used as MPW assembler input.  If that is
003995      the goal, then the caller must convert strings of the form $XX or $XXXX
003996      in the Operand string to their decimal equivalent.  The routine
003997      ModifyOperand is provided in this unit to aid with the conversion
003998      process.
003999
004000      Since a PC-relative comment is an address, the only address that the
004001      Disassembler knows about is the address of the code pointed to by
004002      FirstByte.  Generally, that may be a buffer that has no relation to
004003      "reality", i.e., the actual code loaded into the buffer.  Therefore,
004004      to allow the address comment to be mapped back to some actual address
004005      the caller may specify an adjustment factor, specified by DstAdjust,
004006      that is ADDED to the value that normally would be placed in the
004007      comment.
004008
004009      Operand effective address strings are generated as a function of the
004010      effective address mode and a special case is made for A-trap opcode
004011      strings. In places where a possible symbolic reference could be
004012      substituted for an address (or a portion of an address), the Disassembler
004013      can call a user specified routine to do the substitution (using the
004014      LookupProc parameter described later).  The following table summarizes
004015      the generated effective addresses and where symbolic substitutions (S)
```

```
004016      can be made:
004017
004018      Mode    Generated Effective Address  Effective Address with Substitution
004019      ========================================================================
004020          0     Dn                           Dn
004021          1     An                           An
004022          2     (An)                         (An)
004023          3     (An)+                        (An)+
004024          4     -(An)                        -(An)
004025          5     ∂(An)                        S(An) or just S (if An=A5, ∂≥0)
004026         6n     ∂(An,Xn.Size*Scale)          S(An,Xn.Size*Scale)
004027         6n     (BD,An,Xn.Size*Scale)        (S,An,Xn.Size*Scale)
004028         6n     ([BD,An],Xm.Size*Scale,OD)   ([S,An],Xm.Size*Scale,OD)
004029         6n     ([BD,An,Xn.Size*Scale],OD)   ([S,An,Xn.Size*Scale],OD)
004030         70     ∂                            S
004031         71     ∂                            S
004032         72     *±∂                          S
004033         73     *±∂(Xn.Size*Scale)           S(Xn.Size*Scale)
004034         73     (*±∂,Xn.Size*Scale)          (S,Xn.Size*Scale)
004035         73     ([*±∂],Xm.Size*Scale,OD)     ([S],Xm.Size*Scale,OD)
004036         73     ([*±∂,Xn.Size*Scale],OD)     ([S,Xn.Size*Scale],OD)
004037         74     #data                        S (#data made comment)
004038      A-traps $AXXX                        S (as opcode, AXXX made comment)
004039      ========================================================================
004040
004041      For A-traps, the substitution can be performed to substitute for the DC.W
004042      opcode string.  If the substitution is made then the Disassembler will
004043      generate ,Sys and/or ,Immed flags as operands for Toolbox traps and
004044      ,AutoPop for OS traps when the bits in the trap word indicates these
004045      settings.
004046
004047                      |         Generated          |            Substituted
004048                      | Opcode  Operand  Comment   | Opcode  Operand        Comment
004049      ========================================================================
004050      Toolbox | DC.W    $AXXX    ; TB XXXX | S       [,Sys][,Immed] ; AXXX
004051      OS      | DC.W    $AXXX    ; OS XXXX | S       [,AutoPop]     ; AXXX
004052      ========================================================================
004053
004054      All displacements (∂, BD, OD) are hexadecimal values shown as a byte
004055      ($XX), word ($XXXX), or long ($XXXXXXXX) as appropriate.  The *Scale is
004056      suppressed if 1. The Size is W or L.  Note that effective address
004057      substitutions can only be made for "∂(An)", "BD,An", and "*±∂" cases.
004058
004059      For all the effective address modes 5, 6n, 7n, and for A-traps, a
004060      coroutine (a procedure) whose address is specified by the LookupProc
004061      parameter is called by the Disassembler (if LookupProc is not NIL) to
004062      do the substitution (or A-trap comment) with a string returned by the
004063      proc.  It is assumed that the proc pointed to by LookupProc is a level 1
004064      Pascal proc declared as follows:
004065
004066      PROCEDURE Lookup(      PC:      UNIV Ptr;     {Addr of extension/trap word}
004067                                                BaseReg: LookupRegs;   {Base register/lookup mode  }
004068                                                Opnd:    UNIV LongInt; {Trap word, PC addr, disp.  }
004069                                     VAR S:       Str255);        {Returned substitution      }
004070
004071      or in C,
```

```
004072
004073      pascal void LookUp(Ptr         PC,      /* Addr of extension/trap word */
004074                                        LookupRegs  BaseReg, /* Base register/lookup mode   */
004075                                        long        Opnd,    /* Trap word, PC addr, disp.   */
004076                                        char              *S);     /* Returned substitution       */
004077
004078      PC      = Pointer to instruction extension word or A-trap word in the
004079                        buffer pointed to by the Disassembler's FirstByte parameter.
004080
004081      BaseReg = This determines the meaning of the Opnd value and supplies
004082                        the base register for the "∂(An)", "BD,An", and "*±∂" cases.
004083                        BaseReg may contain any one of the following values:
004084
004085                        _A0_    =  0 ==> A0
004086                        _A1_    =  1 ==> A1
004087                        _A2_    =  2 ==> A2
004088                        _A3_    =  3 ==> A3
004089                        _A4_    =  4 ==> A4
004090                        _A5_    =  5 ==> A5
004091                        _A6_    =  6 ==> A6
004092                        _A7_    =  7 ==> A7
004093                        _PC_    =  8 ==> PC-relative (special case)
004094                        _ABS_   =  9 ==> Abs addr    (special case)
004095                        _TRAP_  = 10 ==> Trap word   (special case)
004096                        _IMM_   = 11 ==> Immediate   (special case)
004097
004098                        For absolute addressing (modes 70 and 71), BaseReg contains
004099                        _ABS_.  For A-traps, BaseReg would contain _TRAP_.  For
004100                        immediate data (mode 74), BaseReg would contain _IMM_.
004101
004102      Opnd    = The contents of this LongInt is determined by the BaseReg
004103                        parameter just described.
004104
004105                        For BaseReg = _IMM_ (immediate data):
004106                                Opnd contains the (extended) 32-bit immediate data specified
004107                                by the instruction.
004108
004109                        For BaseReg = _TRAP_ (A-traps):
004110                                Opnd is the entire trap word. The high order 16 bits of
004111                                Opnd are zero.
004112
004113                        For BaseReg = _ABS_  (absolute effective address):
004114                                Opnd contains the (extended) 32-bit address specifed by
004115                                the instruction's effective address.  Such addresses would
004116                                generally be used to reference low memory globals on a
004117                                Macintosh.
004118
004119                        For BaseReg = _PC_  (PC-relative effective address):
004120                                Opnd contains the 32-bit address represented by "*±∂"
004121                                adjusted by the Disassembler's DstAdjust parameter.
004122
004123                        For BaseReg = _An_  (effective address with a base register):
004124                                Opnd contains the (sign-extended) 32-bit (base)
004125                                displacement from the instruction's effective address.
004126
004127                                In the Macintosh environment, a BaseReg specifying A5
```

```
004128                                  implies either global data references or Jump Table
004129                                  references. Positive Opnd values with an A5 BaseReg thus
004130                                  mean Jump Table references, while a negative offset would
004131                                  mean a global data reference.  Base registers of A6 or A7
004132                                  would usually mean local data.
004133
004134        S       = Pascal string returned from Lookup containing the effective
004135                            address substitution string or a trap name for A-traps.  S is
004136                            set to null PRIOR to calling Lookup.  If it is still null on
004137                            return, the string is not used.  If not null, then for A-traps,
004138                            the returned string is used as the opcode string. In all other
004139                            cases the string is substituted as shown in the above table.
004140
004141        Depending on the application, the caller has three choices on how to
004142        use the Disassembler and an associated Lookup proc:
004143
004144        (1). The caller can call just the Disassembler and provide his own Lookup
004145                    proc. In that case the calling conventions discussed above must be
004146                    followed.
004147
004148        (2). The caller can provide NIL for the LookupProc parameter, in which
004149                    case, NO Lookup proc will be called.
004150
004151        (3). The caller can call first InitLookup (described below, a proc
004152                    provided with this unit) and pass the address of this unit's
004153                    standard Lookup proc when Disassembler is called.   In this case all
004154                    the control logic to determine the kind of substitution to be done
004155                    is provided for the caller and all that need to be provided by the
004156                    user are routines to look up any or all of the following:
004157
004158                    • PC-relative references
004159                    • Jump Table references
004160                    • Absolute address references
004161                    • Trap names
004162                    • Immediate data names
004163                    • References with offsets from base registers                                   *)
004164
004165
004166  PROCEDURE InitLookup(PCRelProc, JTOffProc, TrapProc, AbsAddrProc, IdProc, ImmDataProc: UNIV Ptr);
004167      {Prepare for use of this unit's Lookup proc.  When Disassembler is called
004168       and the address of this unit's Lookup proc is specified, then for immediate
004169       data, PC-relative, Jump Table references, A-traps, absolute addresses, and
004170       offsets from a base register, the associated level 1 Pascal proc
004171       specified here is called (if not NIL -- all six addresses are preset to
004172       NIL).  The calls assume the following declarations for these procs (see
004173       Lookup, below for further details):
004174
004175       PROCEDURE PCRelProc(Address: UNIV LongInt;
004176                                               VAR S:   UNIV Str255);
004177
004178       PROCEDURE JTOffProc(A5JTOffset: UNIV Integer;
004179                                               VAR S:      UNIV Str255);
004180
004181       PROCEDURE TrapNameProc(TrapWord: UNIV Integer;
004182                                                    VAR S:    UNIV Str255);
004183
```

```
004184       PROCEDURE AbsAddrProc(AbsAddr: UNIV LongInt;
004185                                              VAR S:   UNIV Str255);
004186
004187       PROCEDURE IdProc(BaseReg: LookupRegs;
004188                                         Offset:  UNIV LongInt;
004189                                         VAR S:   UNIV Str255);
004190
004191       PROCEDURE ImmDataProc(ImmData: UNIV LongInt;
004192                                              VAR S:   UNIV Str255);
004193
004194       Note: InitLookup contains initialized data which requires initializing
004195                   at load time (this is of concern only to users with assembler
004196                   main programs.}
004197
004198
004199    PROCEDURE Lookup(      PC:      UNIV Ptr;     {Addr of extension/trap word}
004200                                                BaseReg: LookupRegs;   {Base register/lookup mode  }
004201                                                Opnd:    UNIV LongInt; {Trap word, PC addr, disp.  }
004202                                      VAR S:       Str255);        {Returned substitution      }
004203       {This is a standard Lookup proc available to the caller for calls to the
004204        Disassembler.  If the caller elects to use this proc, then InitLookup
004205        MUST be called prior to any calls to the Disassembler.  All the logic
004206        to determine the type of lookup is done by this proc.  For PC-relative,
004207        Jump Table references, A-traps, absolute addresses, and offsets from a
004208        base register, the associated level 1 Pascal proc specified in the
004209        InitLookup call (if not NIL) is called.
004210
004211        This scheme simplifies the Lookup mechanism by allowing the caller
004212        to deal with just the problems related to the application.}
004213
004214
004215    PROCEDURE LookupTrapName(TrapWord: UNIV Integer;
004216                                              VAR S:    UNIV Str255);
004217       {This is a procedure provided to allow conversion of a trap instruction
004218        (in TrapWord) to its corresponding trap name (in S).  It is provided
004219        primarily for use with the Disassembler and its address may be passed to
004220        InitLookup above for use by this unit's Lookup routine.  Alternatively,
004221        there is nothing prohibiting the caller from using it directly for other
004222        purposes or by some other Lookup proc.
004223
004224        Note: The tables in this proc make the size of this proc about 9500
004225                    bytes.  The trap names are fully spelled out in upper and lower
004226                    case.}
004227
004228    PROCEDURE ModifyOperand(VAR Operand: UNIV Str255);
004229       {Scan an operand string, i.e., the null terminated Pascal string returned
004230        by the Disassembler (null MUST be present here) and modify negative hex
004231        values to negated positive value.  For example, $FFFF(A5) would be
004232        modified to -$0001(A5).  The operand to be processed is    passed as the
004233        function's parameter which is edited "in place" and returned to the
004234        caller.
004235
004236        This routine is essentially a pattern matcher and attempts to only
004237        modify 2, 4, and 8 digit hex strings in the operand that "might" be
004238        offsets from a base register.  If the matching tests are passed, the
004239        same number of original digits are output (because that indicates a
```

```
004240       value's size -- byte, word, or long).
004241
004242       For a hex string to be modified, the following tests must be passed:
004243
004244       • There must have been exactly 2, 4, or 8 digits.
004245
004246          Only hex strings $XX, $XXXX, and $XXXXXXXX are possible candidates
004247          because that is the only way the Disassembler generates offsets.
004248
004249       • Hex string must be delimited by a "(" or a ",".
004250
004251          The "(" allows offsets for $XXXX(An,...) and $XX(An,Xn) addressing
004252          modes.  The comma allows for the MC68020 addressing forms.
004253
004254       • The "$X..." must NOT be preceded by a "±".
004255
004256          This eliminates the possibility of modifying the offset of a
004257          PC-relative addressing mode always generated in the form "*±$XXXX".
004258
004259       • The "$X..." must NOT be preceded by a "#".
004260
004261          This eliminates modifying immediate data.
004262
004263       • Value must be negative.
004264
004265          Negative values are the only values we modify.  A value $FFFF is
004266          modified to -$0001.}
004267
004268   FUNCTION validMacsBugSymbol(symStart, limit: UNIV Ptr;
004269                                                    symbol: StringPtr): StringPtr; C;
004270       {Check that the bytes pointed to by symStart represents a valid MacsBug
004271        symbol.  The symbol must be fully contained in the bytes starting at
004272        symStart, up to, but not including, the byte pointed to by the limit
004273        parameter.
004274
004275        If a valid symbol is NOT found, then NIL is returned as the function's
004276        result.  However, if a valid symbol is found, it is copied to symbol (if
004277        it is not NIL) as a null terminated Pascal string, and return a pointer
004278        to where we think the FOLLOWING module begins. In the "old style" cases
004279        (see below) this will always be 8 or 16 bytes after the input symStart.
004280        For new style Apple Pascal and C cases this will depend on the symbol
004281        length, existence of a pad byte, and size of the constant (literal) area.
004282        In all cases, trailing blanks are removed from the symbol.
004283
004284        A valid MacsBug symbol consists of the characters '_', '%', spaces,
004285        digits, and upper/lower case letters in a format determined by the first
004286        two bytes of the symbol as follows:
004287
004288           1st byte  | 2nd byte  |  Byte  |
004289               Range   |  Range    | Length | Comments
004290        ========================================================================
004291           $20 - $7F | $20 - $7F |    8   | "Old style" MacsBug symbol format
004292           $A0 - $7F | $20 - $7F |    8   | "Old style" MacsBug symbol format
004293        ------------------------------------------------------------------------
004294           $20 - $7F | $80 - $FF |   16   | "Old style" MacApp symbol ab==>b.a
004295           $A0 - $7F | $80 - $FF |   16   | "Old style" MacApp symbol ab==>b.a
```

```
004296       ------------------------------------------------------------------------
004297          $80       | $01 - $FF |    n   | n = 2nd byte           (Apple symbol)
004298          $81 - $9F | $00 - $FF |    m   | m = BAnd(1st byte,$7F) (Apple symbol)
004299       ========================================================================
004300
004301       The formats are determined by whether bit 7 is set in the first and
004302       second bytes.  This bit will removed when we find it or'ed into the first
004303       and/or second valid symbol characters.
004304
004305       The first two formats in the above table are the basic "old style" (pre-
004306       existing) MacsBug formats. The first byte may or may not have bit 7 set
004307       the second byte is a valid symbol character.  The first byte (with bit 7
004308       removed) and the next 7 bytes are assumed to comprise the symbol.
004309
004310       The second pair of formats are also "old style" formats, but used for
004311       MacApp symbols.  Bit 7 set in the second character indicates these
004312       formats.  The symbol is assumed to be 16 bytes with the second 8 bytes
004313       preceding the first 8 bytes in the generated symbol.  For example,
004314       12345678abcdefgh represents the symbol abcdefgh.12345678.
004315
004316       The last pair of formats are reserved by Apple and generated by the MPW
004317       Pascal and C compilers.  In these cases the value of the first byte is
004318       always between $80 and $9F, or with bit 7 removed, between $00 and $1F.
004319       For $00, the second byte is the length of the symbol with that many bytes
004320       following the second byte (thus a max length of 255). Values $01 to $1F
004321       represent the length itself.  A pad byte may follow these variable length
004322       cases if the symbol does not end on a word boundary.  Following the
004323       symbol and the possible pad byte is a word containing the size of the
004324       constants (literals) generated by the compiler.
004325
004326       Note that if symStart actually does point to a valid MacsBug symbol,
004327       then you may use showMacsBugSymbol to convert the MacsBug symbol bytes to
004328       a string that could be used as a DC.B operand for disassembly purposes.
004329       This string explicitly shows the MacsBug symbol encodings.}
004330
004331  FUNCTION endOfModule(address, limit: UNIV Ptr; symbol: StringPtr;
004332                                          VAR nextModule: UNIV Ptr): StringPtr; C;
004333      {Check to see if the specified memory address, contains a RTS, JMP (A0) or
004334       RTD #n instruction immediately followed by a valid MacsBug symbol.  These
004335       sequences are the only ones which can determine an end of module when
004336       MacsBug symbols are present.  During the check, the instruction and its
004337       following MacsBug symbol must be fully contained in the bytes starting at
004338       the specified address parameter, up to, but not including, the byte
004339       pointed to by the limit parameter.
004340
004341       If the end of module is NOT found, then NIL is returned as the
004342       function's result.  However, if a end of module is found, the MacsBug
004343       symbol is returned in symbol (if it is not NIL) as a null terminated
004344       Pascal string (with trailing blanks removed), and the functions returns
004345       the pointer to the start of the MacsBug symbol (i.e., address+2 for RTS
004346       or JMP (A0) and address+4 for RTD #n).  This address may then be used as
004347       in input parameter to showMacsBugSymbol to convert the MacsBug symbol to
004348       a Disassembler operand string.
004349
004350       Also returned in nextModule is where we think the FOLLOWING module
004351       begins. In the "old style" cases (see validMacsBugSymbol) this will
```

```
004352       always be 8 or 16 bytes after the input address.  For new style the
004353       Apple Pascal and C cases this will depend on the symbol length, existence
004354       of a pad byte, and size of the constant (literal) area.  See
004355       validMacsBugSymbol for a description of valid MacsBug symbol formats.}
004356
004357  FUNCTION showMacsBugSymbol(symStart, limit: UNIV Ptr; operand: StringPtr;
004358                                                    VAR bytesUsed: Integer): StringPtr; C;
004359      {Format a MacsBug symbol as a operand of a DC.B directive.  The first one
004360       or two bytes of the symbol are generated as $80+'c' if they have there
004361       high high bits set.  All other characters are shown as characters in a
004362       string constant.  The pad byte, if present, is one is also shown as $00.
004363
004364       When called, showMacsBugSymbol assumes that symStart is pointing at a
004365       valid MacsBug symbol as validated by the validMacsBugSymbol or
004366       endOfModule routines. As with validMacsBugSymbol, the symbol must be
004367       fully contained in the bytes starting at symStart up to, but not
004368       including, the byte pointed to by the limit parameter.
004369
004370       The string is returned in the 'operand' parameter as a null terminated
004371       Pascal string.  The function also returns a pointer to this string as its
004372       return value (NIL is returned only if the byte pointed to by the limit
004373       parameter is reached prior to processing the entire symbol -- which
004374       should not happen if properly validated).  The number of bytes used for
004375       the symbol is returned in bytesUsed.  Due to the way MacsBug symbols are
004376       encoded, bytesUsed may not necessarily be the same as the length of the
004377       operand string.
004378
004379       A valid MacsBug symbol consists of the characters '_', '%', spaces,
004380       digits, and upper/lower case letters in a format determined by the first
004381       two bytes of the symbol as described in the validMacsBugSymbol routine.}
004382
004383  {$ENDC}    { UsingDisAsmLookup }
004384
004385  {$IFC NOT UsingIncludes}
004386      END.
004387  {$ENDC}
004388
004389
004390  ### END OF FILE DisAsmLookup.p
004391
```

```
004392
004393  ############################################################################
004394  ### FILE: DiskInit.p
004395  ############################################################################
004396
004397  {
004398  Created: Sunday, January 6, 1991 at 10:30 PM
004399      DiskInit.p
004400      Pascal Interface to the Macintosh Libraries
004401
004402          Copyright Apple Computer, Inc.     1985-1989
004403          All rights reserved
004404  }
004405
004406
004407  {$IFC UNDEFINED UsingIncludes}
004408  {$SETC UsingIncludes := 0}
004409  {$ENDC}
004410
004411  {$IFC NOT UsingIncludes}
004412      UNIT DiskInit;
004413      INTERFACE
004414  {$ENDC}
004415
004416  {$IFC UNDEFINED UsingDiskInit}
004417  {$SETC UsingDiskInit := 1}
004418
004419  {$I+}
004420  {$SETC DiskInitIncludes := UsingIncludes}
004421  {$SETC UsingIncludes := 1}
004422  {$IFC UNDEFINED UsingTypes}
004423  {$I $$Shell(PInterfaces)Types.p}
004424  {$ENDC}
004425  {$SETC UsingIncludes := DiskInitIncludes}
004426
004427  TYPE
004428  HFSDefaults = RECORD
004429      sigWord: PACKED ARRAY [0..1] OF Byte;   { signature word}
004430      abSize: LONGINT;                        { allocation block size in bytes}
004431      clpSize: LONGINT;                       { clump size in bytes}
004432      nxFreeFN: LONGINT;                      { next free file number}
004433      btClpSize: LONGINT;                     { B-Tree clump size in bytes}
004434      rsrv1: INTEGER;                         { reserved}
004435      rsrv2: INTEGER;                         { reserved}
004436      rsrv3: INTEGER;                         { reserved}
004437      END;
004438
004439
004440  PROCEDURE DILoad;
004441  PROCEDURE DIUnload;
004442  FUNCTION DIBadMount(where: Point;evtMessage: LONGINT): INTEGER;
004443  FUNCTION DIFormat(drvNum: INTEGER): OSErr;
004444  FUNCTION DIVerify(drvNum: INTEGER): OSErr;
004445  FUNCTION DIZero(drvNum: INTEGER;volName: Str255): OSErr;
004446
004447
```

```
004448  {$ENDC}    { UsingDiskInit }
004449
004450  {$IFC NOT UsingIncludes}
004451      END.
004452  {$ENDC}
004453
004454
004455  ### END OF FILE DiskInit.p
004456
```

```
004457
004458  ############################################################################
004459  ### FILE: Disks.p
004460  ############################################################################
004461
004462  {
004463  Created: Sunday, January 6, 1991 at 10:30 PM
004464      Disks.p
004465      Pascal Interface to the Macintosh Libraries
004466
004467          Copyright Apple Computer, Inc.     1985-1989
004468          All rights reserved
004469  }
004470
004471
004472  {$IFC UNDEFINED UsingIncludes}
004473  {$SETC UsingIncludes := 0}
004474  {$ENDC}
004475
004476  {$IFC NOT UsingIncludes}
004477      UNIT Disks;
004478      INTERFACE
004479  {$ENDC}
004480
004481  {$IFC UNDEFINED UsingDisks}
004482  {$SETC UsingDisks := 1}
004483
004484  {$I+}
004485  {$SETC DisksIncludes := UsingIncludes}
004486  {$SETC UsingIncludes := 1}
004487  {$IFC UNDEFINED UsingTypes}
004488  {$I $$Shell(PInterfaces)Types.p}
004489  {$ENDC}
004490  {$IFC UNDEFINED UsingOSUtils}
004491  {$I $$Shell(PInterfaces)OSUtils.p}
004492  {$ENDC}
004493  {$SETC UsingIncludes := DisksIncludes}
004494
004495  TYPE
004496  DriveKind = (sony,hard20);
004497
004498
004499  DrvSts = RECORD
004500      track: INTEGER;                 {current track}
004501      writeProt: SignedByte;          {bit 7 = 1 if volume is locked}
004502      diskInPlace: SignedByte;        {disk in drive}
004503      installed: SignedByte;          {drive installed}
004504      sides: SignedByte;              {-1 for 2-sided, 0 for 1-sided}
004505      driveQLink: QElemPtr;           {next queue entry}
004506      driveQVers: INTEGER;            {1 for HD20}
004507      dQDrive: INTEGER;               {drive number}
004508      dQRefNum: INTEGER;              {driver reference number}
004509      dQFSID: INTEGER;                {file system ID}
004510      CASE DriveKind OF
004511        sony:
004512          (twoSideFmt: SignedByte;    {after 1st rd/wrt: 0=1 side, -1=2 side}
```

```
004513          needsFlush: SignedByte;     {-1 for MacPlus drive}
004514          diskErrs: INTEGER);         {soft error count}
004515        hard20:
004516          (driveSize: INTEGER;        {drive block size low word}
004517          driveS1: INTEGER;           {drive block size high word}
004518          driveType: INTEGER;         {1 for HD20}
004519          driveManf: INTEGER;         {1 for Apple Computer, Inc.}
004520          driveChar: SignedByte;      {230 ($E6) for HD20}
004521          driveMisc: SignedByte);     {0 -- reserved}
004522      END;
004523
004524
004525  FUNCTION DiskEject(drvNum: INTEGER): OSErr;
004526  FUNCTION SetTagBuffer(buffPtr: Ptr): OSErr;
004527  FUNCTION DriveStatus(drvNum: INTEGER;VAR status: DrvSts): OSErr;
004528
004529
004530  {$ENDC}    { UsingDisks }
004531
004532  {$IFC NOT UsingIncludes}
004533      END.
004534  {$ENDC}
004535
004536
004537  ### END OF FILE Disks.p
004538
```

```
004539
004540  ############################################################################
004541  ### FILE: Editions.p
004542  ############################################################################
004543
004544  {
004545  Created: Tuesday, January 29, 1991 at 6:35 PM
004546      Editions.p
004547      Pascal Interface to the Macintosh Libraries
004548
004549          Copyright Apple Computer, Inc.    1989-1990
004550          All rights reserved
004551  }
004552
004553
004554  {$IFC UNDEFINED UsingIncludes}
004555  {$SETC UsingIncludes := 0}
004556  {$ENDC}
004557
004558  {$IFC NOT UsingIncludes}
004559      UNIT Editions;
004560      INTERFACE
004561  {$ENDC}
004562
004563  {$IFC UNDEFINED UsingEditions}
004564  {$SETC UsingEditions := 1}
004565
004566  {$I+}
004567  {$SETC EditionsIncludes := UsingIncludes}
004568  {$SETC UsingIncludes := 1}
004569  {$IFC UNDEFINED UsingMemory}
004570  {$I $$Shell(PInterfaces)Memory.p}
004571  {$ENDC}
004572  {$IFC UNDEFINED UsingTypes}
004573  {$I $$Shell(PInterfaces)Types.p}
004574  {$ENDC}
004575  {$IFC UNDEFINED UsingFiles}
004576  {$I $$Shell(PInterfaces)Files.p}
004577  {$ENDC}
004578  {$IFC UNDEFINED UsingAliases}
004579  {$I $$Shell(PInterfaces)Aliases.p}
004580  {$ENDC}
004581  {$SETC UsingIncludes := EditionsIncludes}
004582
004583  CONST
004584
004585  { resource types  }
004586  rSectionType = 'sect';                      { ResType of saved SectionRecords }
004587
004588  { section types }
004589  stSubscriber = $01;
004590  stPublisher = $0A;
004591
004592  sumAutomatic = 0;                           { subscriber update mode - Automatically     }
004593  sumManual = 1;                              { subscriber update mode - Manually }
004594  pumOnSave = 0;                              { publisher update mode - OnSave            }
```

```
004595  pumManual = 1;                              { publisher update mode - Manually }
004596
004597  kPartsNotUsed = 0;
004598  kPartNumberUnknown = -1;                    { misc }
004599
004600  kPreviewWidth = 120;
004601  kPreviewHeight = 120;
004602
004603  kPublisherDocAliasFormat = 'alis';
004604  kPreviewFormat = 'prvw';
004605  kFormatListFormat = 'fmts';
004606
004607  { bits for formatsMask }
004608  kPICTformatMask = 1;
004609  kTEXTformatMask = 2;
004610  ksndFormatMask = 4;
004611
004612  { Finder types for edition files }
004613  kPICTEditionFileType = 'edtp';
004614  kTEXTEditionFileType = 'edtt';
004615  ksndEditionFileType = 'edts';
004616  kUnknownEditionFileType = 'edtu';
004617
004618  { pseudo-item hits for dialogHooks
004619   the first if for NewPublisher or NewSubscriber Dialogs }
004620  emHookRedrawPreview = 150;
004621
004622  { the following are for SectionOptions Dialog }
004623  emHookCancelSection = 160;
004624  emHookGoToPublisher = 161;
004625  emHookGetEditionNow = 162;
004626  emHookSendEditionNow = 162;
004627  emHookManualUpdateMode = 163;
004628  emHookAutoUpdateMode = 164;
004629
004630  { the refcon field of the dialog record during a modalfilter
004631   or dialoghook contains one the following }
004632  emOptionsDialogRefCon = 'optn';
004633  emCancelSectionDialogRefCon = 'cncl';
004634  emGoToPubErrDialogRefCon = 'gerr';
004635  kFormatLengthUnknown = -1;
004636
004637  TYPE
004638  SectionType = SignedByte;                   { one byte, stSubscriber or stPublisher }
004639  TimeStamp = LONGINT;                        { seconds since 1904 }
004640  FormatType = PACKED ARRAY [1..4] OF CHAR;   { similar to ResType as used by scrap mgr }
004641  EditionRefNum = Handle;                     { used in Edition I/O }
004642  { update modes }
004643  UpdateMode = INTEGER;                       { sumAutomatic, pumSuspend, etc }
004644
004645  SectionPtr = ^SectionRecord;
004646  SectionHandle = ^SectionPtr;
004647  SectionRecord = RECORD
004648      version: SignedByte;                    { always 0x01 in system 7.0 }
004649      kind: SectionType;                      { stSubscriber or stPublisher }
004650      mode: UpdateMode;                       { auto or manual }
```

```
004651      mdDate: TimeStamp;                      { last change in document }
004652      sectionID: LONGINT;                     { app. specific, unique per document }
004653      refCon: LONGINT;                        { application specific }
004654      alias: AliasHandle;                     { handle to Alias Record }
004655      subPart: LONGINT;                       { which part of container file }
004656      nextSection: SectionHandle;             { for linked list of app's Sections }
004657      controlBlock: Handle;                   { used internally }
004658      refNum: EditionRefNum;                  { used internally }
004659      END;
004660
004661  EditionContainerSpecPtr = ^EditionContainerSpec;
004662  EditionContainerSpec = RECORD
004663      theFile: FSSpec;
004664      theFileScript: ScriptCode;
004665      thePart: LONGINT;
004666      thePartName: Str31;
004667      thePartScript: ScriptCode;
004668      END;
004669
004670  EditionInfoRecord = RECORD
004671      crDate: TimeStamp;                      { date EditionContainer was created }
004672      mdDate: TimeStamp;                      { date of last change }
004673      fdCreator: OSType;                      { file creator }
004674      fdType: OSType;                         { file type }
004675      container: EditionContainerSpec;        { the Edition }
004676      END;
004677
004678  NewPublisherReply = RECORD
004679      canceled: BOOLEAN;                      { O }
004680      replacing : BOOLEAN;
004681      usePart: BOOLEAN;                       { I }
004682      preview: Handle;                        { I }
004683      previewFormat: FormatType;              { I }
004684      container: EditionContainerSpec;        { I/O }
004685      END;
004686
004687  NewSubscriberReply = RECORD
004688      canceled: BOOLEAN;                      { O }
004689      formatsMask: SignedByte;
004690      container: EditionContainerSpec;        {I/O}
004691      END;
004692
004693  SectionOptionsReply = RECORD
004694      canceled: BOOLEAN;                      { O }
004695      changed: BOOLEAN;                       { O }
004696      sectionH: SectionHandle;                { I }
004697      action: ResType;                        { O }
004698      END;
004699
004700
004701  ExpModalFilterProcPtr = ProcPtr;            { FUNCTION Filter(theDialog: DialogPtr; VAR theEvent: EventRecord; itemOffset: INTEGER; VAR itemHit: I
004702  ExpDlgHookProcPtr = ProcPtr;                { FUNCTION Hook(itemOffset, item: INTEGER; theDialog: DialogPtr; yourDataPtr: Ptr): INTEGER; }
004703
004704  FormatIOVerb = (ioHasFormat,ioReadFormat,ioNewFormat,ioWriteFormat);
004705
004706
```

```
004707  FormatIOParamBlock = RECORD
004708      ioRefNum: LONGINT;
004709      format: FormatType;
004710      formatIndex: LONGINT;
004711      offset: LONGINT;
004712      buffPtr: Ptr;
004713      buffLen: LONGINT;
004714      END;
004715
004716
004717  FormatIOProcPtr = ProcPtr;                  { FUNCTION IO(selector: FormatIOVerb; VAR PB: FormatIOParamBlock): OSErr; }
004718
004719  EditionOpenerVerb = (eoOpen,eoClose,eoOpenNew,eoCloseNew,eoCanSubscribe);
004720
004721
004722  EditionOpenerParamBlock = RECORD
004723      info: EditionInfoRecord;
004724      sectionH: SectionHandle;
004725      document: FSSpecPtr;
004726      fdCreator: OSType;
004727      ioRefNum: LONGINT;
004728      ioProc: FormatIOProcPtr;
004729      success: BOOLEAN;
004730      formatsMask: SignedByte;
004731      END;
004732
004733
004734  EditionOpenerProcPtr = ProcPtr;             { FUNCTION Opener(selector: EditionOpenerVerb; VAR PB: EditionOpenerParamBlock): OSErr; }
004735
004736  CONST
004737
004738  {
004739      Section events now arrive in the message buffer using the AppleEvent format.
004740      The direct object parameter is an aeTemporaryIDParamType ('tid '). The temporary
004741      ID's type is rSectionType ('sect') and the 32-bit value is a SectionHandle.
004742      The following is a sample buffer
004743
004744      name            offset    contents
004745      ----            ------    --------
004746
004747      header            0        'aevt'
004748      majorVersion    4        0x01
004749      minorVersion    6        0x01
004750      endOfMetaData    8        ';;;;'
004751      directObjKey    12        '----'
004752      paramType        16        'tid '
004753      paramLength        20        0x0008
004754      tempIDType        24        'sect'
004755      tempID             28        the SectionHandle <-- this is want you want}
004756
004757  sectionEventMsgClass = 'sect';
004758  sectionReadMsgID = 'read';
004759  sectionWriteMsgID = 'writ';
004760  sectionScrollMsgID = 'scrl';
004761  sectionCancelMsgID = 'cncl';
004762
```

```
004763  FUNCTION InitEditionPack: OSErr;
004764      INLINE $3F3C,$0011,$303C,$0100,$A82D;
004765  FUNCTION NewSection(container: EditionContainerSpec;
004766                      sectionDocument: FSSpecPtr;
004767                      kind: SectionType;
004768                      sectionID: LONGINT;
004769                      initalMode: UpdateMode;
004770                      VAR sectionH: SectionHandle): OSErr;
004771      INLINE $303C,$0A02,$A82D;
004772  FUNCTION RegisterSection(sectionDocument: FSSpec;
004773                           sectionH: SectionHandle;
004774                           VAR aliasWasUpdated: BOOLEAN): OSErr;
004775      INLINE $303C,$0604,$A82D;
004776  FUNCTION UnRegisterSection(sectionH: SectionHandle): OSErr;
004777      INLINE $303C,$0206,$A82D;
004778  FUNCTION IsRegisteredSection(sectionH: SectionHandle): OSErr;
004779      INLINE $303C,$0208,$A82D;
004780  FUNCTION AssociateSection(sectionH: SectionHandle;
004781                            newSectionDocument: FSSpecPtr): OSErr;
004782      INLINE $303C,$040C,$A82D;
004783  FUNCTION CreateEditionContainerFile(editionFile: FSSpec;
004784                                      fdCreator: OSType;
004785                                      editionFileNameScript: ScriptCode): OSErr;
004786      INLINE $303C,$050E,$A82D;
004787  FUNCTION DeleteEditionContainerFile(editionFile: FSSpec): OSErr;
004788      INLINE $303C,$0210,$A82D;
004789  FUNCTION OpenEdition(subscriberSectionH: SectionHandle;
004790                       VAR refNum: EditionRefNum): OSErr;
004791      INLINE $303C,$0412,$A82D;
004792  FUNCTION OpenNewEdition(publisherSectionH: SectionHandle;
004793                          fdCreator: OSType;
004794                          publisherSectionDocument: FSSpecPtr;
004795                          VAR refNum: EditionRefNum): OSErr;
004796      INLINE $303C,$0814,$A82D;
004797  FUNCTION CloseEdition(whichEdition: EditionRefNum;
004798                        successful: BOOLEAN): OSErr;
004799      INLINE $303C,$0316,$A82D;
004800  FUNCTION EditionHasFormat(whichEdition: EditionRefNum;
004801                            whichFormat: FormatType;
004802                            VAR formatSize: Size): OSErr;
004803      INLINE $303C,$0618,$A82D;
004804  FUNCTION ReadEdition(whichEdition: EditionRefNum;
004805                       whichFormat: FormatType;
004806                       buffPtr: UNIV Ptr;
004807                       VAR buffLen: Size): OSErr;
004808      INLINE $303C,$081A,$A82D;
004809  FUNCTION WriteEdition(whichEdition: EditionRefNum;
004810                        whichFormat: FormatType;
004811                        buffPtr: UNIV Ptr;
004812                        buffLen: Size): OSErr;
004813      INLINE $303C,$081C,$A82D;
004814  FUNCTION GetEditionFormatMark(whichEdition: EditionRefNum;
004815                                whichFormat: FormatType;
004816                                VAR currentMark: LONGINT): OSErr;
004817      INLINE $303C,$061E,$A82D;
004818  FUNCTION SetEditionFormatMark(whichEdition: EditionRefNum;
```

```
004819                              whichFormat: FormatType;
004820                              setMarkTo: LONGINT): OSErr;
004821      INLINE $303C,$0620,$A82D;
004822  FUNCTION GetEditionInfo(sectionH: SectionHandle;
004823                          VAR editionInfo: EditionInfoRecord): OSErr;
004824      INLINE $303C,$0422,$A82D;
004825  FUNCTION GoToPublisherSection(container: EditionContainerSpec): OSErr;
004826      INLINE $303C,$0224,$A82D;
004827  FUNCTION GetLastEditionContainerUsed(VAR container: EditionContainerSpec): OSErr;
004828      INLINE $303C,$0226,$A82D;
004829  FUNCTION GetStandardFormats(container: EditionContainerSpec;
004830                              VAR previewFormat: FormatType;
004831                              preview: Handle;
004832                              publisherAlias: Handle;
004833                              formats: Handle): OSErr;
004834      INLINE $303C,$0A28,$A82D;
004835  FUNCTION GetEditionOpenerProc(VAR opener: EditionOpenerProcPtr): OSErr;
004836      INLINE $303C,$022A,$A82D;
004837  FUNCTION SetEditionOpenerProc(opener: EditionOpenerProcPtr): OSErr;
004838      INLINE $303C,$022C,$A82D;
004839  FUNCTION CallEditionOpenerProc(selector: EditionOpenerVerb;
004840                                 VAR PB: EditionOpenerParamBlock;
004841                                 routine: EditionOpenerProcPtr): OSErr;
004842      INLINE $303C,$052E,$A82D;
004843  FUNCTION CallFormatIOProc(selector: FormatIOVerb;
004844                            VAR PB: FormatIOParamBlock;
004845                            routine: FormatIOProcPtr): OSErr;
004846      INLINE $303C,$0530,$A82D;
004847  FUNCTION NewSubscriberDialog(VAR reply: NewSubscriberReply): OSErr;
004848      INLINE $303C,$0232,$A82D;
004849  FUNCTION NewSubscriberExpDialog(VAR reply: NewSubscriberReply;
004850                                  where: Point;
004851                                  expansionDITLresID: INTEGER;
004852                                  dlgHook: ExpDlgHookProcPtr;
004853                                  filterProc: ExpModalFilterProcPtr;
004854                                  yourDataPtr:UNIV Ptr): OSErr;
004855      INLINE $303C,$0B34,$A82D;
004856  FUNCTION NewPublisherDialog(VAR reply: NewPublisherReply): OSErr;
004857      INLINE $303C,$0236,$A82D;
004858  FUNCTION NewPublisherExpDialog(VAR reply: NewPublisherReply;
004859                                 where: Point;
004860                                 expansionDITLresID: INTEGER;
004861                                 dlgHook: ExpDlgHookProcPtr;
004862                                 filterProc: ExpModalFilterProcPtr;
004863                                 yourDataPtr: UNIV Ptr): OSErr;
004864      INLINE $303C,$0B38,$A82D;
004865  FUNCTION SectionOptionsDialog(VAR reply: SectionOptionsReply): OSErr;
004866      INLINE $303C,$023A,$A82D;
004867  FUNCTION SectionOptionsExpDialog(VAR reply: SectionOptionsReply;
004868                                   where: Point;
004869                                   expansionDITLresID: INTEGER;
004870                                   dlgHook: ExpDlgHookProcPtr;
004871                                   filterProc: ExpModalFilterProcPtr;
004872                                   yourDataPtr: UNIV Ptr): OSErr;
004873      INLINE $303C,$0B3C,$A82D;
004874
```

```
004875
004876  {$ENDC}    { UsingEditions }
004877
004878  {$IFC NOT UsingIncludes}
004879      END.
004880  {$ENDC}
004881
004882
004883  ### END OF FILE Editions.p
004884
```

```
004885
004886  ############################################################################
004887  ### FILE: ENET.p
004888  ############################################################################
004889
004890
004891  {
004892  Created: Sunday, September 15, 1991 at 10:23 PM
004893   ENET.p
004894   Pascal Interface to the Macintosh Libraries
004895
004896    Copyright Apple Computer, Inc. 1990-1991
004897    All rights reserved
004898  }
004899
004900
004901  {$IFC UNDEFINED UsingIncludes}
004902  {$SETC UsingIncludes := 0}
004903  {$ENDC}
004904
004905  {$IFC NOT UsingIncludes}
004906   UNIT ENET;
004907   INTERFACE
004908  {$ENDC}
004909
004910  {$IFC UNDEFINED UsingENET}
004911  {$SETC UsingENET := 1}
004912
004913  {$I+}
004914  {$SETC ENETIncludes := UsingIncludes}
004915  {$SETC UsingIncludes := 1}
004916  {$IFC UNDEFINED UsingTypes}
004917  {$I $$Shell(PInterfaces)Types.p}
004918  {$ENDC}
004919  {$IFC UNDEFINED UsingOSUtils}
004920  {$I $$Shell(PInterfaces)OSUtils.p}
004921  {$ENDC}
004922  {$SETC UsingIncludes := ENETIncludes}
004923
004924  CONST
004925  ENetSetGeneral = 253;                       {Set "general" mode}
004926  ENetGetInfo = 252;                          {Get info}
004927  ENetRdCancel = 251;                         {Cancel read}
004928  ENetRead = 250;                             {Read}
004929  ENetWrite = 249;                            {Write}
004930  ENetDetachPH = 248;                         {Detach protocol handler}
004931  ENetAttachPH = 247;                         {Attach protocol handler}
004932  ENetAddMulti = 246;                         {Add a multicast address}
004933  ENetDelMulti = 245;                         {Delete a multicast address}
004934
004935  eLenErr = -92;                              {Length error ddpLenErr}
004936  eMultiErr = -91;                            {Multicast address error ddpSktErr}
004937
004938  EAddrRType = 'eadr';                        {Alternate address resource type}
004939
004940  TYPE
```

```
004941  EParamBlkPtr = ^EParamBlock;
004942  EParamBlock = PACKED RECORD
004943   qLink: QElemPtr;                          {next queue entry}
004944   qType: INTEGER;                           {queue type}
004945   ioTrap: INTEGER;                          {routine trap}
004946   ioCmdAddr: Ptr;                           {routine address}
004947   ioCompletion: ProcPtr;                    {completion routine}
004948   ioResult: OSErr;                          {result code}
004949   ioNamePtr: StringPtr;                     {->filename}
004950   ioVRefNum: INTEGER;                       {volume reference or drive number}
004951   ioRefNum: INTEGER;                        {driver reference number}
004952   csCode: INTEGER;                          {call command code AUTOMATICALLY set}
004953   CASE INTEGER OF
004954     ENetWrite,ENetAttachPH,ENetDetachPH,ENetRead,ENetRdCancel,ENetGetInfo,ENetSetGeneral:
004955    (eProtType: INTEGER;                     {Ethernet protocol type}
004956    ePointer: Ptr;
004957    eBuffSize: INTEGER;                      {buffer size}
004958    eDataSize: INTEGER);                     {number of bytes read}
004959     ENetAddMulti,ENetDelMulti:
004960    (eMultiAddr: ARRAY [0..5] of char);       {Multicast Address}
004961   END;
004962
004963
004964  FUNCTION EWrite(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004965  FUNCTION EAttachPH(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004966  FUNCTION EDetachPH(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004967  FUNCTION ERead(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004968  FUNCTION ERdCancel(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004969  FUNCTION EGetInfo(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004970  FUNCTION ESetGeneral(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004971  FUNCTION EAddMulti(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004972  FUNCTION EDelMulti(thePBptr: EParamBlkPtr;async: BOOLEAN): OSErr;
004973
004974
004975  {$ENDC} { UsingENET }
004976
004977  {$IFC NOT UsingIncludes}
004978   END.
004979  {$ENDC}
004980
004981
004982  ### END OF FILE ENET.p
004983
```

```
004984
004985  ############################################################################
004986  ### FILE: EPPC.p
004987  ############################################################################
004988
004989  {
004990  Created: Sunday, January 6, 1991 at 10:32 PM
004991      EPPC.p
004992      Pascal Interface to the Macintosh Libraries
004993
004994          Copyright Apple Computer, Inc.    1988-1990
004995          All rights reserved
004996  }
004997
004998
004999  {$IFC UNDEFINED UsingIncludes}
005000  {$SETC UsingIncludes := 0}
005001  {$ENDC}
005002
005003  {$IFC NOT UsingIncludes}
005004      UNIT EPPC;
005005      INTERFACE
005006  {$ENDC}
005007
005008  {$IFC UNDEFINED UsingEPPC}
005009  {$SETC UsingEPPC := 1}
005010
005011  {$I+}
005012  {$SETC EPPCIncludes := UsingIncludes}
005013  {$SETC UsingIncludes := 1}
005014  {$IFC UNDEFINED UsingPPCToolbox}
005015  {$I $$Shell(PInterfaces)PPCToolbox.p}
005016  {$ENDC}
005017  {$IFC UNDEFINED UsingProcesses}
005018  {$I $$Shell(PInterfaces)Processes.p}
005019  {$ENDC}
005020  {$IFC UNDEFINED UsingEvents}
005021  {$I $$Shell(PInterfaces)Events.p}
005022  {$ENDC}
005023  {$SETC UsingIncludes := EPPCIncludes}
005024
005025  CONST
005026  kHighLevelEvent = 23;
005027
005028  { postOptions currently supported }
005029  receiverIDMask = $0000F000;
005030  receiverIDisPSN = $00008000;
005031  receiverIDisSignature = $00007000;
005032  receiverIDisSessionID = $00006000;
005033  receiverIDisTargetID = $00005000;
005034
005035  systemOptionsMask = $00000F00;
005036  nReturnReceipt = $00000200;
005037
005038  priorityMask = $000000FF;
005039  nAttnMsg = $00000001;
```

```
005040
005041  { error returns from Post and Accept }
005042
005043  bufferIsSmall = -607;
005044  noOutstandingHLE = -608;
005045  connectionInvalid = -609;
005046  noUserInteractionAllowed = -610;        { no user interaction allowed }
005047
005048  { constant for return receipts }
005049
005050  HighLevelEventMsgClass = 'jaym';
005051  rtrnReceiptMsgID = 'rtrn';
005052  msgWasPartiallyAccepted = 2;
005053  msgWasFullyAccepted = 1;
005054  msgWasNotAccepted = 0;
005055
005056  TYPE
005057  TargetIDPtr = ^TargetID;
005058  TargetIDHdl = ^TargetIDPtr;
005059  TargetID = RECORD
005060      sessionID: LONGINT;
005061      name: PPCPortRec;
005062      location: LocationNameRec;
005063      recvrName: PPCPortRec;
005064      END;
005065
005066
005067  SenderID = TargetID;
005068  SenderIDPtr = ^SenderID;
005069
005070  HighLevelEventMsgPtr = ^HighLevelEventMsg;
005071  HighLevelEventMsgHdl = ^HighLevelEventMsgPtr;
005072  HighLevelEventMsg = RECORD
005073      HighLevelEventMsgHeaderLength: INTEGER;
005074      version: INTEGER;
005075      reserved1: LONGINT;
005076      theMsgEvent: EventRecord;
005077      userRefcon: LONGINT;
005078      postingOptions: LONGINT;
005079      msgLength: LONGINT;
005080      END;
005081
005082
005083  FUNCTION PostHighLevelEvent(theEvent: EventRecord;
005084                              receiverID: Ptr;
005085                              msgRefcon: LONGINT;
005086                              msgBuff: Ptr;
005087                              msgLen: LONGINT;
005088                              postingOptions: LONGINT): OSErr;
005089      INLINE $3F3C,$0034,$A88F;
005090  FUNCTION AcceptHighLevelEvent(VAR sender: TargetID;
005091                                VAR msgRefcon: LONGINT;
005092                                msgBuff: Ptr;
005093                                VAR msgLen: LONGINT): OSErr;
005094      INLINE $3F3C,$0033,$A88F;
005095  FUNCTION GetProcessSerialNumberFromPortName(portName: PPCPortRec;VAR PSN: ProcessSerialNumber): OSErr;
```

```
005096      INLINE $3F3C,$0035,$A88F;
005097  FUNCTION GetPortNameFromProcessSerialNumber(VAR portName: PPCPortRec;PSN: ProcessSerialNumber): OSErr;
005098      INLINE $3F3C,$0046,$A88F;
005099
005100  TYPE
005101  GetSpecificFilterProcPtr = ProcPtr;     { FUNCTION MyFilter(yourDataPtr: Ptr;               }
005102                                          {                   msgBuff: HighLevelEventMsgPtr;    }
005103                                          {                   sender: TargetID): Boolean;        }
005104
005105  FUNCTION GetSpecificHighLevelEvent(aFilter: GetSpecificFilterProcPtr;yourDataPtr: UNIV Ptr;
005106      VAR err: OSErr): BOOLEAN;
005107      INLINE $3F3C,$0045,$A88F;
005108
005109
005110  {$ENDC}    { UsingEPPC }
005111
005112  {$IFC NOT UsingIncludes}
005113      END.
005114  {$ENDC}
005115
005116
005117  ### END OF FILE EPPC.p
005118
```

```
005119
005120  ************************************************************************
005121  *** FILE: ErrMgr.p
005122  ************************************************************************
005123
005124  {
005125  Created: Tuesday, August 2, 1988 at 12:26 PM
005126      ErrMgr.p
005127      Pascal Interface to the Macintosh Libraries
005128
005129
005130      <<< ErrMgr - Error File Manager Routines Interface File >>>
005131
005132
005133      Copyright Apple Computer, Inc.  1987-1988
005134      All rights reserved
005135
005136      This file contains:
005137
005138      InitErrMgr(toolname, sysename, Nbrs)  - ErrMgr initialization
005139      CloseErrMgr()                         - Close ErrMgr message files
005140      GetSysErrText(Nbr, Msg)               - Get a system error message for a number
005141      GetToolErrText(Nbr, Insert, Msg)      - Get a tool error message for a number
005142      AddErrInsert(insert, msgString)       - Add an insert to a message
005143
005144  }
005145
005146
005147  {$IFC UNDEFINED UsingIncludes}
005148  {$SETC UsingIncludes := 0}
005149  {$ENDC}
005150
005151  {$IFC NOT UsingIncludes}
005152      UNIT ErrMgr;
005153      INTERFACE
005154  {$ENDC}
005155
005156  {$IFC UNDEFINED UsingErrMgr}
005157  {$SETC UsingErrMgr := 1}
005158
005159  {$I+}
005160  {$SETC ErrMgrIncludes := UsingIncludes}
005161  {$SETC UsingIncludes := 1}
005162  {$IFC UNDEFINED UsingTypes}
005163  {$I $$Shell(PInterfaces)Types.p}
005164  {$ENDC}
005165  {$SETC UsingIncludes := ErrMgrIncludes}
005166
005167
005168  PROCEDURE InitErrMgr(toolErrFilename: Str255;sysErrFilename: Str255;showToolErrNbrs: BOOLEAN);
005169  { ErrMgr initialization.This must be done before using any other ErrMgr
005170  routine.  Set showToolErrNbrs to true if you want all tool messages to contain
005171  the error number following the message text enclosed in parentheses (e.g.,
005172  "<msg txt> ([OS] Error <n>)"; system error messages always contain the error
005173  number).  The toolErrFileName parameter is used to specify the name of a
005174  tool-specific error file, and should be the NULL or a null string if not used
```

```
005175  (or if the tool's data fork is to be used as the error file, see
005176  GetToolErrText for futher details). The sysErrFileName parameter is used to
005177  specify the name of a system error file, and should normally be the NULL or a
005178  null string, which causes the ErrMgr to look in the MPW Shell directory for
005179  "SysErrs.Err" (see GetSysErrText). }
005180
005181  PROCEDURE CloseErrMgr; C;
005182  { Ideally a CloseErrMgr should be done at the end of execution to make sure all
005183  files opened by the ErrMgr are closed.  You can let normal program termination
005184  do the closing.  But if you are a purist...
005185   }
005186
005187  PROCEDURE GetSysErrText(msgNbr: INTEGER;errMsg: StringPtr);
005188  (* Get the error message text corresponding to system error number errNbr from
005189  the system error message file (whose name was specified in the InitErrMgr
005190  call).  The text of the message is returned in errMsg and the function returns
005191  a pointer to errMsg.  The maximum length of the message is limited to 254
005192  characters.
005193
005194  Note, if a system message filename was not specified to InitErrMgr, then the
005195  ErrMgr assumes the message file contained in the file "SysErrs.Err".  This
005196  file is first accessed as "                                {ShellDirectory}SysErrs.Err" on the assumption that
005197  SysErrs.Err is kept in the same directory as the MPW Shell.  If the file
005198  cannot be opened, then an open is attempted on "SysErrs.Err" in the System
005199  Folder. *)
005200
005201  PROCEDURE AddErrInsert(insert: Str255;msgString: StringPtr); C;
005202  { Add another insert to an error message string.This call is used when more
005203  than one insert is to be added to a message (because it contains more than
005204  one '^' character).
005205   }
005206
005207  PROCEDURE GetToolErrText(msgNbr: INTEGER;errInsert: Str255;errMsg: StringPtr);
005208  (* Get the error message text corresponding to tool error number errNbr from
005209  the tool error message file (whose name was specified in the InitErrMgr
005210  call).  The text of the message is returned in errMsg and the function returns
005211  a pointer to errMsg.  The maximum length of the message is limited to 254
005212  characters.  If the message is to have an insert, then ErrInsert should be a
005213  pointer to it.  Otherwise it should be either be a null string or a NULL
005214  pointer.
005215
005216  Inserts are indicated in error messages by specifying a '^' to indicate where
005217  the insert is to be placed.
005218
005219  Note, if a tool message filename was not specified to InitErrMgr, then the
005220  ErrMgr assumes the message file contained in the data fork of the tool calling
005221  the ErrMgr.  This name is contained in the Shell variable   {Command} and the
005222  value of that variable is used to open the error message file. *)
005223
005224
005225
005226  {$ENDC}    { UsingErrMgr }
005227
005228  {$IFC NOT UsingIncludes}
005229      END.
005230  {$ENDC}
```

```
005231
005232
005233  ### END OF FILE ErrMgr.p
005234
```

```
005235
005236  ##########################################################################
005237  ### FILE: Errors.p
005238  ##########################################################################
005239
005240  {
005241  Created: Thursday, March 14, 1991 at 4:02 PM
005242      Errors.p
005243      Pascal Interface to the Macintosh Libraries
005244
005245          Copyright Apple Computer, Inc.    1985-1990
005246          All rights reserved
005247  }
005248
005249
005250  {$IFC UNDEFINED UsingIncludes}
005251  {$SETC UsingIncludes := 0}
005252  {$ENDC}
005253
005254  {$IFC NOT UsingIncludes}
005255      UNIT Errors;
005256      INTERFACE
005257  {$ENDC}
005258
005259  {$IFC UNDEFINED UsingErrors}
005260  {$SETC UsingErrors := 1}
005261
005262
005263  CONST
005264  paramErr = -50;                             {error in user parameter list}
005265  noHardwareErr = -200;                       {Sound Manager Error Returns}
005266  notEnoughHardwareErr = -201;                {Sound Manager Error Returns}
005267  userCanceledErr = -128;
005268  qErr = -1;                                  {queue element not found during deletion}
005269  vTypErr = -2;                               {invalid queue element}
005270  corErr = -3;                                {core routine number out of range}
005271  unimpErr = -4;                              {unimplemented core routine}
005272  SlpTypeErr = -5;                            {invalid queue element}
005273  seNoDB = -8;                                {no debugger installed to handle debugger
005274  command}
005275  controlErr = -17;                           {I/O System Errors}
005276  statusErr = -18;                            {I/O System Errors}
005277  readErr = -19;                              {I/O System Errors}
005278  writErr = -20;                              {I/O System Errors}
005279  badUnitErr = -21;                           {I/O System Errors}
005280  unitEmptyErr = -22;                         {I/O System Errors}
005281  openErr = -23;                              {I/O System Errors}
005282  closErr = -24;                              {I/O System Errors}
005283  dRemovErr = -25;                            {tried to remove an open driver}
005284  dInstErr = -26;                             {DrvrInstall couldn't find driver in
005285  resources }
005286  abortErr = -27;                             {IO call aborted by KillIO}
005287  iIOAbortErr = -27;                          {IO abort error (Printing Manager)}
005288  notOpenErr = -28;                           {Couldn't rd/wr/ctl/sts cause driver not
005289  opened}
005290  unitTblFullErr = -29;                       {unit table has no more entries}
```

```
005291  dceExtErr = -30;                        {dce extension error}
005292  slotNumErr = -360;                      {invalid slot # error}
005293  gcrOnMFMErr = -400;                     {gcr format on high density media error}
005294  dirFulErr = -33;                        {Directory full}
005295  dskFulErr = -34;                        {disk full}
005296  nsvErr = -35;                           {no such volume}
005297  ioErr = -36;                            {I/O error (bummers)}
005298  bdNamErr = -37;                         {there may be no bad names in the final
005299  system!}
005300  fnOpnErr = -38;                         {File not open}
005301  eofErr = -39;                           {End of file}
005302  posErr = -40;                           {tried to position to before start of file
005303  (r/w)}
005304  mFulErr = -41;                          {memory full (open) or file won't fit
005305  (load)}
005306  tmfoErr = -42;                          {too many files open}
005307  fnfErr = -43;                           {File not found}
005308  wPrErr = -44;                           {diskette is write protected.}
005309  fLckdErr = -45;                         {file is locked}
005310  vLckdErr = -46;                         {volume is locked}
005311  fBsyErr = -47;                          {File is busy (delete)}
005312  dupFNErr = -48;                         {duplicate filename (rename)}
005313  opWrErr = -49;                          {file already open with with write
005314  permission}
005315  rfNumErr = -51;                         {refnum error}
005316  gfpErr = -52;                           {get file position error}
005317  volOffLinErr = -53;                     {volume not on line error (was Ejected)}
005318  permErr = -54;                          {permissions error (on file open)}
005319  volOnLinErr = -55;                      {drive volume already on-line at MountVol}
005320  nsDrvErr = -56;                         {no such drive (tried to mount a bad drive
005321  num)}
005322  noMacDskErr = -57;                      {not a mac diskette (sig bytes are wrong)}
005323  extFSErr = -58;                         {volume in question belongs to an external
005324  fs}
005325  fsRnErr = -59;                          {file system internal error:during rename
005326  the old entry was deleted but could not be restored.}
005327  badMDBErr = -60;                        {bad master directory block}
005328  wrPermErr = -61;                        {write permissions error}
005329  dirNFErr = -120;                        {Directory not found}
005330  tmwdoErr = -121;                        {No free WDCB available}
005331  badMovErr = -122;                       {Move into offspring error}
005332  wrgVolTypErr = -123;                    {Wrong volume type error [operation not
005333  supported for MFS]}
005334  volGoneErr = -124;                      {Server volume has been disconnected.}
005335  fidNotFound = -1300;                    {no file thread exists.}
005336  fidExists = -1301;                      {file id already exists}
005337  notAFileErr = -1302;                    {directory specified}
005338  diffVolErr = -1303;                     {files on different volumes}
005339  catChangedErr = -1304;                  {the catalog has been modified}
005340  desktopDamagedErr = -1305;              {desktop database files are corrupted}
005341  sameFileErr = -1306;                    {can't exchange a file with itself}
005342  badFidErr = -1307;                      {file id is dangling or doesn't match with
005343  the file number}
005344  envNotPresent = -5500;                  {returned by glue.}
005345  envBadVers = -5501;                     {Version non-positive}
005346  envVersTooBig = -5502;                  {Version bigger than call can handle}
```

```
005347  fontDecError = -64;                      {error during font declaration}
005348  fontNotDeclared = -65;                   {font not declared}
005349  fontSubErr = -66;                        {font substitution occured}
005350  fontNotOutlineErr = -32615;              {bitmap font passed to routine that does
005351  outlines only}
005352  firstDskErr = -84;                       {I/O System Errors}
005353  lastDskErr = -64;                        {I/O System Errors}
005354  noDriveErr = -64;                        {drive not installed}
005355  offLinErr = -65;                         {r/w requested for an off-line drive}
005356  noNybErr = -66;                          {couldn't find 5 nybbles in 200 tries}
005357  noAdrMkErr = -67;                        {couldn't find valid addr mark}
005358  dataVerErr = -68;                        {read verify compare failed}
005359  badCksmErr = -69;                        {addr mark checksum didn't check}
005360  badBtSlpErr = -70;                       {bad addr mark bit slip nibbles}
005361  noDtaMkErr = -71;                        {couldn't find a data mark header}
005362  badDCksum = -72;                         {bad data mark checksum}
005363  badDBtSlp = -73;                         {bad data mark bit slip nibbles}
005364  wrUnderrun = -74;                        {write underrun occurred}
005365  cantStepErr = -75;                       {step handshake failed}
005366  tk0BadErr = -76;                         {track 0 detect doesn't change}
005367  initIWMErr = -77;                        {unable to initialize IWM}
005368  twoSideErr = -78;                        {tried to read 2nd side on a 1-sided
005369  drive}
005370  spdAdjErr = -79;                         {unable to correctly adjust disk speed}
005371  seekErr = -80;                           {track number wrong on address mark}
005372  sectNFErr = -81;                         {sector number never found on a track}
005373  fmt1Err = -82;                           {can't find sector 0 after track format}
005374  fmt2Err = -83;                           {can't get enough sync}
005375  verErr = -84;                            {track failed to verify}
005376  clkRdErr = -85;                          {unable to read same clock value twice}
005377  clkWrErr = -86;                          {time written did not verify}
005378  prWrErr = -87;                           {parameter ram written didn't read-verify}
005379  prInitErr = -88;                         {InitUtil found the parameter ram
005380  uninitialized}
005381  rcvrErr = -89;                           {SCC receiver error (framing; parity; OR)}
005382  breakRecd = -90;                         {Break received (SCC)}
005383
005384  {Power Manager Errors}
005385  pmBusyErr = -13000;                      {Power Mgr never ready to start handshake}
005386  pmReplyTOErr = -13001;                   {Timed out waiting for reply}
005387  pmSendStartErr = -13002;                 {during send, pmgr did not start hs}
005388  pmSendEndErr = -13003;                   {during send, pmgr did not finish hs}
005389  pmRecvStartErr = -13004;                 {during receive, pmgr did not start hs}
005390  pmRecvEndErr = -13005;                   {during receive, pmgr did not finish hs
005391  configured for this connection}
005392
005393  {Scrap Manager errors}
005394  noScrapErr = -100;                       {No scrap exists error}
005395  noTypeErr = -102;                        {No object of that type in scrap}
005396  memROZWarn = -99;                        {soft error in ROZ}
005397  memROZError = -99;                       {hard error in ROZ}
005398  memROZErr = -99;                         {hard error in ROZ}
005399  memFullErr = -108;                       {Not enough room in heap zone}
005400  nilHandleErr = -109;                     {Master Pointer was NIL in HandleZone or
005401  other}
005402  memWZErr = -111;                         {WhichZone failed (applied to free block)}
```

```
005403  memPurErr = -112;                           {trying to purge a locked or non-purgeable
005404  block}
005405  memAdrErr = -110;                           {address was odd; or out of range}
005406  memAZErr = -113;                            {Address in zone check failed}
005407  memPCErr = -114;                            {Pointer Check failed}
005408  memBCErr = -115;                            {Block Check failed}
005409  memSCErr = -116;                            {Size Check failed}
005410  memLockedErr = -117;                        {trying to move a locked block (MoveHHi)}
005411  resNotFound = -192;                         {Resource not found}
005412  resFNotFound = -193;                        {Resource file not found}
005413  addResFailed = -194;                        {AddResource failed}
005414  addRefFailed = -195;                        {AddReference failed}
005415  rmvResFailed = -196;                        {RmveResource failed}
005416  rmvRefFailed = -197;                        {RmveReference failed}
005417  resAttrErr = -198;                          {attribute inconsistent with operation}
005418  mapReadErr = -199;                          {map inconsistent with operation}
005419  CantDecompress = -186;                      {resource bent ("the bends") - can't
005420  decompress a compressed resource}
005421  badExtResource = -185;                      {extended resource has a bad format.}
005422  evtNotEnb = 1;                              {event not enabled at PostEvent}
005423  noMemForPictPlaybackErr = -145;
005424  rgnTooBigError = -147;
005425  pixMapTooDeepErr = -148;
005426  nsStackErr = -149;
005427  cMatchErr = -150;                           {Color2Index failed to find an index}
005428  cTempMemErr = -151;                         {failed to allocate memory for temporary
005429  structures}
005430  cNoMemErr = -152;                           {failed to allocate memory for structure}
005431  cRangeErr = -153;                           {range error on colorTable request}
005432  cProtectErr = -154;                         {colorTable entry protection violation}
005433  cDevErr = -155;                             {invalid type of graphics device}
005434  cResErr = -156;                             {invalid resolution for MakeITable}
005435  rgnTooBigErr = -500;
005436  updPixMemErr = -125;                        {insufficient memory to update a pixmap}
005437  pictInfoVersionErr = -11000;                { wrong version of the PictInfo structure
005438  }
005439  pictInfoIDErr = -11001;                     { the internal consistancy check for the
005440  PictInfoID is wrong }
005441  pictInfoVerbErr = -11002;                   { the passed verb was invalid }
005442  cantLoadPickMethodErr = -11003;             { unable to load the custom pick proc }
005443  colorsRequestedErr = -11004;                { the number of colors requested was
005444  illegal }
005445  pictureDataErr = -11005;                    { the picture data was invalid }
005446
005447  {Sound Manager errors}
005448  noHardware = noHardwareErr;                 {*** obsolete spelling}
005449  notEnoughHardware = notEnoughHardwareErr;   {*** obsolete spelling}
005450  queueFull = -203;                           {Sound Manager Error Returns}
005451  resProblem = -204;                          {Sound Manager Error Returns}
005452  badChannel = -205;                          {Sound Manager Error Returns}
005453  badFormat = -206;                           {Sound Manager Error Returns}
005454  notEnoughBufferSpace = -207;                { could not allocate enough memory }
005455  badFileFormat = -208;                       { was not type AIFF or was of bad
005456  format,corrupt }
005457  channelBusy = -209;                         { the Channel is being used for a PFD
005458  already }
```

```
005459  buffersTooSmall = -210;                 { can not operate in the memory allowed }
005460  channelNotBusy = -211;
005461  noMoreRealTime = -212;                  { not enough CPU cycles left to add
005462  another task }
005463  siNoSoundInHardware = -220;             {no Sound Input hardware}
005464  siBadSoundInDevice = -221;              {invalid index passed to
005465  SoundInGetIndexedDevice}
005466  siNoBufferSpecified = -222;             {returned by synchronous SPBRecord if nil
005467  buffer passed}
005468  siInvalidCompression = -223;            {invalid compression type}
005469  siHardDriveTooSlow = -224;              {hard drive too slow to record to disk}
005470  siInvalidSampleRate = -225;             {invalid sample rate}
005471  siInvalidSampleSize = -226;             {invalid sample size}
005472  siDeviceBusyErr = -227;                 {input device already in use}
005473  siBadDeviceName = -228;                 {input device could not be opened}
005474  siBadRefNum = -229;                     {invalid input device reference number}
005475  siInputDeviceErr = -230;                {input device hardware failure}
005476  siUnknownInfoType = -231;               {invalid info type selector (returned by
005477  driver)}
005478  siUnknownQuality = -232;                {invalid quality selector (returned by
005479  driver)}
005480
005481  {Notification Manager errors}
005482  nmTypErr = -299;                        {wrong queue type}
005483  siInitSDTblErr = 1;                     {slot int dispatch table could not be
005484  initialized.}
005485  siInitVBLQsErr = 2;                     {VBLqueues for all slots could not be
005486  initialized.}
005487  siInitSPTblErr = 3;                     {slot priority table could not be
005488  initialized.}
005489  sdmJTInitErr = 10;                      {SDM Jump Table could not be initialized.}
005490  sdmInitErr = 11;                        {SDM could not be initialized.}
005491  sdmSRTInitErr = 12;                     {Slot Resource Table could not be
005492  initialized.}
005493  sdmPRAMInitErr = 13;                    {Slot PRAM could not be initialized.}
005494  sdmPriInitErr = 14;                     {Cards could not be initialized.}
005495  smSDMInitErr = -290;                    {Error; SDM could not be initialized.}
005496  smSRTInitErr = -291;                    {Error; Slot Resource Table could not be
005497  initialized.}
005498  smPRAMInitErr = -292;                   {Error; Slot Resource Table could not be
005499  initialized.}
005500  smPriInitErr = -293;                    {Error; Cards could not be initialized.}
005501  smEmptySlot = -300;                     {No card in slot}
005502  smCRCFail = -301;                       {CRC check failed for declaration data}
005503  smFormatErr = -302;                     {FHeader Format is not Apple's}
005504  smRevisionErr = -303;                   {Wrong revison level}
005505  smNoDir = -304;                         {Directory offset is Nil }
005506  smDisabledSlot = -305;                  {This slot is disabled (-305 use to be
005507  smLWTstBad)}
005508  smNosInfoArray = -306;                  {No sInfoArray. Memory Mgr error.}
005509  smResrvErr = -307;                      {Fatal reserved error. Resreved field <>
005510  0.}
005511  smUnExBusErr = -308;                    {Unexpected BusError}
005512  smBLFieldBad = -309;                    {ByteLanes field was bad.}
005513  smFHBlockRdErr = -310;                  {Error occured during _sGetFHeader.}
005514  smFHBlkDispErr = -311;                  {Error occured during _sDisposePtr
```

```
005515  (Dispose of FHeader block).}
005516  smDisposePErr = -312;                       {_DisposePointer error}
005517  smNoBoardSRsrc = -313;                      {No Board sResource.}
005518  smGetPRErr = -314;                          {Error occured during _sGetPRAMRec (See
005519  SIMStatus).}
005520  smNoBoardId = -315;                         {No Board Id.}
005521  smInitStatVErr = -316;                      {The InitStatusV field was negative after
005522  primary or secondary init.}
005523  smInitTblVErr = -317;                       {An error occured while trying to
005524  initialize the Slot Resource Table.}
005525  smNoJmpTbl = -318;                          {SDM jump table could not be created.}
005526  smBadBoardId = -319;                        {BoardId was wrong; re-init the PRAM
005527  record.}
005528  smBusErrTO = -320;                          {BusError time out.}
005529
005530  { The following errors are for primary or secondary init code.  The errors are logged
005531  in the
005532  vendor status field of the sInfo record.  Normally the vendor error is not Apple's
005533  concern,
005534  but a special error is needed to patch secondary inits.}
005535
005536
005537  svTempDisable = -32768;                     {Temporarily disable card but run primary
005538  init.}
005539  svDisabled = -32640;                        {Reserve range -32640 to -32768 for Apple
005540  temp disables.}
005541  smBadRefId = -330;                          {Reference Id not found in List}
005542  smBadsList = -331;                          {Bad sList: Id1 < Id2 < Id3 ...format is
005543  not followed.}
005544  smReservedErr = -332;                       {Reserved field not zero}
005545  smCodeRevErr = -333;                        {Code revision is wrong}
005546  smCPUErr = -334;                            {Code revision is wrong}
005547  smsPointerNil = -335;                       {LPointer is nil From sOffsetData. If this
005548  error occurs; check sInfo rec for more information.}
005549  smNilsBlockErr = -336;                      {Nil sBlock error (Dont allocate and try
005550  to use a nil sBlock)}
005551  smSlotOOBErr = -337;                        {Slot out of bounds error}
005552  smSelOOBErr = -338;                         {Selector out of bounds error}
005553  smNewPErr = -339;                           {_NewPtr error}
005554  smBlkMoveErr = -340;                        {_BlockMove error}
005555  smCkStatusErr = -341;                       {Status of slot = fail.}
005556  smGetDrvrNamErr = -342;                     {Error occured during _sGetDrvrName.}
005557  smDisDrvrNamErr = -343;                     {Error occured during _sDisDrvrName.}
005558  smNoMoresRsrcs = -344;                      {No more sResources}
005559  smsGetDrvrErr = -345;                       {Error occurred during _sGetDriver.}
005560  smBadsPtrErr = -346;                        {Bad pointer was passed to sCalcsPointer}
005561  smByteLanesErr = -347;                      {NumByteLanes was determined to be zero.}
005562  smOffsetErr = -348;                         {Offset was too big (temporary error}
005563  smNoGoodOpens = -349;                       {No opens were successfull in the loop.}
005564  smSRTOvrFlErr = -350;                       {SRT over flow.}
005565  smRecNotFnd = -351;                         {Record not found in the SRT.}
005566  editionMgrInitErr = -450;                   {edition manager not inited by this app}
005567  badSectionErr = -451;                       {not a valid SectionRecord}
005568  notRegisteredSectionErr = -452;             {not a registered SectionRecord}
005569  badEditionFileErr = -453;                   {edition file is corrupt}
005570  badSubPartErr = -454;                       {can not use sub parts in this release}
```

```
005571  multiplePublisherWrn = -460;                {A Publisher is already registered for
005572  that container}
005573  containerNotFoundWrn = -461;                {could not find editionContainer at this
005574  time}
005575  containerAlreadyOpenWrn = -462;             {container already opened by this section}
005576  notThePublisherWrn = -463;                  {not the first registered publisher for
005577  that container}
005578  teScrapSizeErr = -501;                      {scrap item too big for text edit record}
005579  hwParamErr = -502;                          {bad selector for _HWPriv}
005580
005581  { Process Manager errors }
005582  procNotFound = -600;                        { no eligible process with specified
005583  descriptor }
005584  memFragErr = -601;                          { not enough room to launch app w/special
005585  requirements }
005586  appModeErr = -602;                          { memory mode is 32-bit, but app not 32-
005587  bit clean }
005588  protocolErr = -603;                         { app made module calls in improper order
005589  }
005590  hardwareConfigErr = -604;                   { hardware configuration not correct for
005591  call }
005592  appMemFullErr = -605;                       { application SIZE not big enough for
005593  launch }
005594  appIsDaemon = -606;                         { app is BG-only, and launch flags
005595  disallow this }
005596
005597  {MemoryDispatch errors}
005598  notEnoughMemoryErr = -620;                  {insufficient physical memory}
005599  notHeldErr = -621;                          {specified range of memory is not held}
005600  cannotMakeContiguousErr = -622;             {cannot make specified range contiguous}
005601  notLockedErr = -623;                        {specified range of memory is not locked}
005602  interruptsMaskedErr = -624;                 {don't call with interrupts masked}
005603  cannotDeferErr = -625;                      {unable to defer additional functions}
005604  ddpSktErr = -91;                            {error in soket number}
005605  ddpLenErr = -92;                            {data length too big}
005606  noBridgeErr = -93;                          {no network bridge for non-local send}
005607  lapProtErr = -94;                           {error in attaching/detaching protocol}
005608  excessCollsns = -95;                        {excessive collisions on write}
005609  portInUse = -97;                            {driver Open error code (port is in use)}
005610  portNotCf = -98;                            {driver Open error code (parameter RAM not
005611  configured for this connection)}
005612  nbpBuffOvr = -1024;                         {Buffer overflow in LookupName}
005613  nbpNoConfirm = -1025;
005614  nbpConfDiff = -1026;                        {Name confirmed at different socket}
005615  nbpDuplicate = -1027;                       {Duplicate name exists already}
005616  nbpNotFound = -1028;                        {Name not found on remove}
005617  nbpNISErr = -1029;                          {Error trying to open the NIS}
005618  aspBadVersNum = -1066;                      {Server cannot support this ASP version}
005619  aspBufTooSmall = -1067;                     {Buffer too small}
005620  aspNoMoreSess = -1068;                      {No more sessions on server}
005621  aspNoServers = -1069;                       {No servers at that address}
005622  aspParamErr = -1070;                        {Parameter error}
005623  aspServerBusy = -1071;                      {Server cannot open another session}
005624  aspSessClosed = -1072;                      {Session closed}
005625  aspSizeErr = -1073;                         {Command block too big}
005626  aspTooMany = -1074;                         {Too many clients (server error)}
```

```
005627  aspNoAck = -1075;                       {No ack on attention request (server err)}
005628  reqFailed = -1096;
005629  tooManyReqs = -1097;
005630  tooManySkts = -1098;
005631  badATPSkt = -1099;
005632  badBuffNum = -1100;
005633  noRelErr = -1101;
005634  cbNotFound = -1102;
005635  noSendResp = -1103;
005636  noDataArea = -1104;
005637  reqAborted = -1105;
005638  buf2SmallErr = -3101;
005639  noMPPErr = -3102;
005640  ckSumErr = -3103;
005641  extractErr = -3104;
005642  readQErr = -3105;
005643  atpLenErr = -3106;
005644  atpBadRsp = -3107;
005645  recNotFnd = -3108;
005646  sktClosedErr = -3109;
005647  afpAccessDenied = -5000;
005648  afpAuthContinue = -5001;
005649  afpBadUAM = -5002;
005650  afpBadVersNum = -5003;
005651  afpBitmapErr = -5004;
005652  afpCantMove = -5005;
005653  afpDenyConflict = -5006;
005654  afpDirNotEmpty = -5007;
005655  afpDiskFull = -5008;
005656  afpEofError = -5009;
005657  afpFileBusy = -5010;
005658  afpFlatVol = -5011;
005659  afpItemNotFound = -5012;
005660  afpLockErr = -5013;
005661  afpMiscErr = -5014;
005662  afpNoMoreLocks = -5015;
005663  afpNoServer = -5016;
005664  afpObjectExists = -5017;
005665  afpObjectNotFound = -5018;
005666  afpParmErr = -5019;
005667  afpRangeNotLocked = -5020;
005668  afpRangeOverlap = -5021;
005669  afpSessClosed = -5022;
005670  afpUserNotAuth = -5023;
005671  afpCallNotSupported = -5024;
005672  afpObjectTypeErr = -5025;
005673  afpTooManyFilesOpen = -5026;
005674  afpServerGoingDown = -5027;
005675  afpCantRename = -5028;
005676  afpDirNotFound = -5029;
005677  afpIconTypeError = -5030;
005678  afpVolLocked = -5031;                   {Volume is Read-Only}
005679  afpObjectLocked = -5032;                {Object is M/R/D/W inhibited}
005680  afpContainsSharedErr = -5033;           {$FFFFEC57   the folder being shared
005681  contains a shared folder }
005682  afpIDNotFound = -5034;                  {$FFFFEC56}
```

```
005683  afpIDExists = -5035;                    {$FFFFEC55}
005684  afpDiffVolErr = -5036;                  {$FFFFEC54}
005685  afpCatalogChanged = -5037;              {$FFFFEC53}
005686  afpSameObjectErr = -5038;               {$FFFFEC52}
005687  afpBadIDErr = -5039;                    {$FFFFEC51}
005688  afpPwdSameErr = -5040;                  {$FFFFEC50  someone tried to change their
005689  password to the same password on a mantadory password change }
005690  afpPwdTooShortErr = -5041;              {$FFFFEC4F  the password being set is too
005691  short: there is a minimum length that must be met or exceeded }
005692  afpPwdExpiredErr = -5042;               {$FFFFEC4E  the password being used is too
005693  old: this requires the user to change the password before log-in can continue }
005694  afpInsideSharedErr = -5043;             {$FFFFEC4D  the folder being shared is
005695  inside a shared folder OR the folder contains a shared folder and is being moved into
005696  a shared folder OR the folder contains a shared folder and is being moved into the
005697  descendent of a shared folder. }
005698  afpInsideTrashErr = -5044;              {$FFFFEC4C  the folder being shared is
005699  inside the trash folder OR the shared folder is being moved into the trash folder OR
005700  the folder is being moved to the trash and it contains a shared folder }
005701
005702  {PPC errors}
005703  notInitErr = -900;                      { PPCToolBox not initialized }
005704  nameTypeErr = -902;                     { Invalid or inappropriate
005705  locationKindSelector in locationName }
005706  noPortErr = -903;                       { Unable to open port or bad portRefNum }
005707  noGlobalsErr = -904;                    { The system is hosed, better re-boot }
005708  localOnlyErr = -905;                    { Network activity is currently disabled }
005709  destPortErr = -906;                     { Port does not exist at destination }
005710  sessTableErr = -907;                    { Out of session tables, try again later }
005711  noSessionErr = -908;                    { Invalid session reference number }
005712  badReqErr = -909;                       { bad parameter or invalid state for
005713  operation }
005714  portNameExistsErr = -910;               { port is already open (perhaps in another
005715  app) }
005716  noUserNameErr = -911;                   { user name unknown on destination machine
005717  }
005718  userRejectErr = -912;                   { Destination rejected the session request
005719  }
005720  noMachineNameErr = -913;                { user hasn't named his Macintosh in the
005721  Network Setup Control Panel }
005722  noToolboxNameErr = -914;                { A system resource is missing, not too
005723  likely }
005724  noResponseErr = -915;                   { unable to contact destination }
005725  portClosedErr = -916;                   { port was closed }
005726  sessClosedErr = -917;                   { session was closed }
005727  badPortNameErr = -919;                  { PPCPortRec malformed }
005728  noDefaultUserErr = -922;                { user hasn't typed in owners name in
005729  Network Setup Control Pannel }
005730  notLoggedInErr = -923;                  { The default userRefNum does not yet
005731  exist }
005732  noUserRefErr = -924;                    { unable to create a new userRefNum }
005733  networkErr = -925;                      { An error has occured in the network, not
005734  too likely }
005735  noInformErr = -926;                     { PPCStart failed because destination did
005736  not have inform pending }
005737  authFailErr = -927;                     { unable to authenticate user at
005738  destination }
```

```
005739  noUserRecErr = -928;                    { Invalid user reference number }
005740  badServiceMethodErr = -930;             { illegal service type, or not supported }
005741  badLocNameErr = -931;                   { location name malformed }
005742  guestNotAllowedErr = -932;              { destination port requires authentication
005743  }
005744  swOverrunErr = 1;                       {serial driver error masks}
005745  parityErr = 16;                         {serial driver error masks}
005746  hwOverrunErr = 32;                      {serial driver error masks}
005747  framingErr = 64;                        {serial driver error masks}
005748  dsBusError = 1;                         {bus error }
005749  dsAddressErr = 2;                       {address error}
005750  dsIllInstErr = 3;                       {illegal instruction error}
005751  dsZeroDivErr = 4;                       {zero divide error}
005752  dsChkErr = 5;                           {check trap error}
005753  dsOvflowErr = 6;                        {overflow trap error}
005754  dsPrivErr = 7;                          {privilege violation error}
005755  dsTraceErr = 8;                         {trace mode error}
005756  dsLineAErr = 9;                         {line 1010 trap error}
005757  dsLineFErr = 10;                        {line 1111 trap error}
005758  dsMiscErr = 11;                         {miscellaneous hardware exception error}
005759  dsCoreErr = 12;                         {unimplemented core routine error}
005760  dsIrqErr = 13;                          {uninstalled interrupt error}
005761  dsIOCoreErr = 14;                       {IO Core Error}
005762  dsLoadErr = 15;                         {Segment Loader Error}
005763  dsFPErr = 16;                           {Floating point error}
005764  dsNoPackErr = 17;                       {package 0 not present}
005765  dsNoPk1 = 18;                           {package 1 not present}
005766  dsNoPk2 = 19;                           {package 2 not present}
005767  dsNoPk3 = 20;                           {package 3 not present}
005768  dsNoPk4 = 21;                           {package 4 not present}
005769  dsNoPk5 = 22;                           {package 5 not present}
005770  dsNoPk6 = 23;                           {package 6 not present}
005771  dsNoPk7 = 24;                           {package 7 not present}
005772  dsMemFullErr = 25;                      {out of memory!}
005773  dsBadLaunch = 26;                       {can't launch file}
005774  dsFSErr = 27;                           {file system map has been trashed}
005775  dsStknHeap = 28;                        {stack has moved into application heap}
005776  negZcbFreeErr = 33;                     {ZcbFree has gone negative}
005777  dsFinderErr = 41;                       {can't load the Finder error}
005778  dsBadSlotInt = 51;                      {unserviceable slot interrupt}
005779  dsBadSANEOpcode = 81;                   {bad opcode given to SANE Pack4}
005780  dsBadPatchHeader = 83;                  {SetTrapAddress saw the “come-from”
005781  header}
005782  menuPrgErr = 84;                        {happens when a menu is purged}
005783  dsMBarNFnd = 85;                        {Menu Manager Errors}
005784  dsHMenuFindErr = 86;                    {Menu Manager Errors}
005785  dsWDEFNotFound = 87;                    {could not load WDEF}
005786  dsCDEFNotFound = 88;                    {could not load CDEF}
005787  dsMDEFNotFound = 89;                    {could not load MDEF}
005788  dsNoFPU = 90;                           {an FPU instruction was executed and the
005789  machine doesn’t have one}
005790  dsNoPatch = 98;                         {Can't patch for particular Model Mac}
005791  dsBadPatch = 99;                        {Can't load patch resource}
005792  dsParityErr = 101;                      {memory parity error}
005793  dsOldSystem = 102;                      {System is too old for this ROM}
005794  ds32BitMode = 103;                      {booting in 32-bit on a 24-bit sys}
```

```
005795  dsNeedToWriteBootBlocks = 104;              {need to write new boot blocks}
005796  dsNotEnoughRAMToBoot = 105;                 {must have at least 1.5MB of RAM to boot
005797  7.0}
005798  dsBufPtrTooLow = 106;                       {bufPtr moved too far during boot}
005799  dsReinsert = 30;                            {request user to reinsert off-line volume}
005800  shutDownAlert = 42;                         {handled like a shutdown error}
005801  dsShutDownOrRestart = 20000;                {user choice between ShutDown and Restart}
005802  dsSwitchOffOrRestart = 20001;               {user choice between switching off and
005803  Restart}
005804  dsForcedQuit = 20002;                       {allow the user to ExitToShell, return if
005805  Cancel}
005806
005807  {System Errors that are used after MacsBug is loaded to put up dialogs since these
005808  should not cause MacsBug to stop, they must be in the range (30, 42, 16384-32767)
005809  negative numbers add to an existing dialog without putting up a whole new dialog}
005810  dsMacsBugInstalled = -10;                   {say “MacsBug Installed”}
005811  dsDisassemblerInstalled = -11;              {say “Disassembler Installed”}
005812  dsExtensionsDisabled = -13;                 {say “Extensions Disabled”}
005813  dsGreeting = 40;                            {welcome to Macintosh greeting}
005814  dsSysErr = 32767;                           {general system error}
005815
005816  {old names here for compatibility’s sake}
005817  WDEFNFnd = dsWDEFNotFound;
005818  CDEFNFnd = dsCDEFNotFound;
005819  dsNotThe1 = 31;                             {not the disk I wanted}
005820  dsBadStartupDisk = 42;                      {unable to mount boot volume (sad Mac
005821  only)}
005822  dsSystemFileErr = 43;                       {can’t find System file to open (sad Mac
005823  only)}
005824  dsHD20Installed = -12;                      {say “HD20 Startup”}
005825  mBarNFnd = -126;                            {system error code for MBDF not found}
005826  hMenuFindErr = -127;                        {could not find HMenu's parent in MenuKey}
005827  userBreak = -490;                           {user debugger break}
005828  strUserBreak = -491;                        {user debugger break; display string on
005829  stack}
005830  exUserBreak = -492;                         {user debugger break; execute debugger
005831  commands on stack}
005832
005833  {obsolete errors that are no longer used, but I don’t have the guts to remove from
005834  this file}
005835  selectorErr = paramErr;                     {bad selector, for selector-based traps}
005836
005837  PROCEDURE SysError(errorCode: INTEGER);
005838      INLINE $301F,$A9C9;
005839
005840
005841  {$ENDC}    { UsingErrors }
005842
005843  {$IFC NOT UsingIncludes}
005844      END.
005845  {$ENDC}
005846
005847
005848  ### END OF FILE Errors.p
005849
```

```
005850
005851  ############################################################################
005852  ### FILE: Events.p
005853  ############################################################################
005854
005855
005856  {
005857  Created: Sunday, September 15, 1991 at 10:40 PM
005858   Events.p
005859   Pascal Interface to the Macintosh Libraries
005860
005861    Copyright Apple Computer, Inc. 1985-1991
005862    All rights reserved
005863  }
005864
005865
005866  {$IFC UNDEFINED UsingIncludes}
005867  {$SETC UsingIncludes := 0}
005868  {$ENDC}
005869
005870  {$IFC NOT UsingIncludes}
005871   UNIT Events;
005872   INTERFACE
005873  {$ENDC}
005874
005875  {$IFC UNDEFINED UsingEvents}
005876  {$SETC UsingEvents := 1}
005877
005878  {$I+}
005879  {$SETC EventsIncludes := UsingIncludes}
005880  {$SETC UsingIncludes := 1}
005881  {$IFC UNDEFINED UsingTypes}
005882  {$I $$Shell(PInterfaces)Types.p}
005883  {$ENDC}
005884  {$IFC UNDEFINED UsingQuickdraw}
005885  {$I $$Shell(PInterfaces)Quickdraw.p}
005886  {$ENDC}
005887  {$SETC UsingIncludes := EventsIncludes}
005888
005889  CONST
005890  nullEvent = 0;
005891  mouseDown = 1;
005892  mouseUp = 2;
005893  keyDown = 3;
005894  keyUp = 4;
005895  autoKey = 5;
005896  updateEvt = 6;
005897  diskEvt = 7;
005898  activateEvt = 8;
005899  osEvt = 15;
005900
005901  { event mask equates }
005902  mDownMask = 2;
005903  mUpMask = 4;
005904  keyDownMask = 8;
005905  keyUpMask = 16;
```

```
005906  autoKeyMask = 32;
005907  updateMask = 64;
005908  diskMask = 128;
005909  activMask = 256;
005910  highLevelEventMask = 1024;
005911  osMask = -32768;
005912  everyEvent = -1;
005913
005914  { event message equates }
005915  charCodeMask = $000000FF;
005916  keyCodeMask = $0000FF00;
005917  adbAddrMask = $00FF0000;
005918  osEvtMessageMask = $FF000000;
005919
005920  { OS event messages.  Event (sub)code is in the high byte of the message field. }
005921  mouseMovedMessage = $FA;
005922  suspendResumeMessage = $01;
005923
005924  resumeFlag = 1;             { Bit 0 of message indicates resume vs suspend }
005925  convertClipboardFlag = 2;   { Bit 1 in resume message indicates clipboard change }
005926
005927  { modifiers }
005928  activeFlag = 1;             { Bit 0 of modifiers for activateEvt and mouseDown events }
005929  btnState = 128;             { Bit 7 of low byte is mouse button state }
005930  cmdKey = 256;               { Bit 0 }
005931  shiftKey = 512;             { Bit 1 }
005932  alphaLock = 1024;           { Bit 2 }
005933  optionKey = 2048;           { Bit 3 of high byte }
005934  controlKey = 4096;
005935
005936  { obsolete equates }
005937  networkEvt = 10;
005938  driverEvt = 11;
005939  app1Evt = 12;
005940  app2Evt = 13;
005941  app3Evt = 14;
005942  app4Evt = 15;
005943  networkMask = 1024;
005944  driverMask = 2048;
005945  app1Mask = 4096;
005946  app2Mask = 8192;
005947  app3Mask = 16384;
005948  app4Mask = -32768;
005949
005950  TYPE
005951  EventRecord = RECORD
005952   what: INTEGER;
005953   message: LONGINT;
005954   when: LONGINT;
005955   where: Point;
005956   modifiers: INTEGER;
005957   END;
005958
005959
005960  KeyMap = PACKED ARRAY [0..127] OF BOOLEAN;
005961
```

```
005962  FUNCTION GetNextEvent(eventMask: INTEGER;VAR theEvent: EventRecord): BOOLEAN;
005963   INLINE $A970;
005964  FUNCTION WaitNextEvent(eventMask: INTEGER;VAR theEvent: EventRecord;sleep: LONGINT;
005965   mouseRgn: RgnHandle): BOOLEAN;
005966   INLINE $A860;
005967  FUNCTION EventAvail(eventMask: INTEGER;VAR theEvent: EventRecord): BOOLEAN;
005968   INLINE $A971;
005969  PROCEDURE GetMouse(VAR mouseLoc: Point);
005970   INLINE $A972;
005971  FUNCTION Button: BOOLEAN;
005972   INLINE $A974;
005973  FUNCTION StillDown: BOOLEAN;
005974   INLINE $A973;
005975  FUNCTION WaitMouseUp: BOOLEAN;
005976   INLINE $A977;
005977  PROCEDURE GetKeys(VAR theKeys: KeyMap);
005978   INLINE $A976;
005979  FUNCTION TickCount: LONGINT;
005980   INLINE $A975;
005981  FUNCTION GetDblTime: LONGINT;
005982   INLINE $2EB8,$02F0;
005983  FUNCTION GetCaretTime: LONGINT;
005984   INLINE $2EB8,$02F4;
005985
005986
005987  {$ENDC} { UsingEvents }
005988
005989  {$IFC NOT UsingIncludes}
005990   END.
005991  {$ENDC}
005992
005993
005994  ### END OF FILE Events.p
005995
```

```
005996
005997  ############################################################################
005998  ### FILE: Files.p
005999  ############################################################################
006000
006001
006002  {
006003  Created: Sunday, September 15, 1991 at 10:42 PM
006004   Files.p
006005   Pascal Interface to the Macintosh Libraries
006006
006007    Copyright Apple Computer, Inc. 1985-1991
006008    All rights reserved
006009  }
006010
006011
006012  {$IFC UNDEFINED UsingIncludes}
006013  {$SETC UsingIncludes := 0}
006014  {$ENDC}
006015
006016  {$IFC NOT UsingIncludes}
006017   UNIT Files;
006018   INTERFACE
006019  {$ENDC}
006020
006021  {$IFC UNDEFINED UsingFiles}
006022  {$SETC UsingFiles := 1}
006023
006024  {$I+}
006025  {$SETC FilesIncludes := UsingIncludes}
006026  {$SETC UsingIncludes := 1}
006027  {$IFC UNDEFINED UsingTypes}
006028  {$I $$Shell(PInterfaces)Types.p}
006029  {$ENDC}
006030  {$IFC UNDEFINED UsingOSUtils}
006031  {$I $$Shell(PInterfaces)OSUtils.p}
006032  {$ENDC}
006033  {$IFC UNDEFINED UsingSegLoad}
006034  {$I $$Shell(PInterfaces)SegLoad.p}
006035  {$ENDC}
006036  {$SETC UsingIncludes := FilesIncludes}
006037
006038  CONST
006039
006040  { Finder Constants }
006041  fsAtMark = 0;
006042  fOnDesk = 1;
006043  fsCurPerm = 0;
006044  fHasBundle = 8192;
006045  fsRdPerm = 1;
006046  fInvisible = 16384;
006047  fTrash = -3;
006048  fsWrPerm = 2;
006049  fDesktop = -2;
006050  fsRdWrPerm = 3;
006051  fDisk = 0;
```

```
006052  fsRdWrShPerm = 4;
006053  fsFromStart = 1;
006054  fsFromLEOF = 2;
006055  fsFromMark = 3;
006056  rdVerify = 64;
006057  ioDirFlg = 3;                               { see IM IV-125 }
006058  ioDirMask = $10;
006059  fsRtParID = 1;
006060  fsRtDirID = 2;
006061
006062  { CatSearch SearchBits Constants }
006063  fsSBPartialName = 1;
006064  fsSBFullName = 2;
006065  fsSBFlAttrib = 4;
006066  fsSBFlFndrInfo = 8;
006067  fsSBFlLgLen = 32;
006068  fsSBFlPyLen = 64;
006069  fsSBFlRLgLen = 128;
006070  fsSBFlRPyLen = 256;
006071  fsSBFlCrDat = 512;
006072  fsSBFlMdDat = 1024;
006073  fsSBFlBkDat = 2048;
006074  fsSBFlXFndrInfo = 4096;
006075  fsSBFlParID = 8192;
006076  fsSBNegate = 16384;
006077  fsSBDrUsrWds = 8;
006078  fsSBDrNmFls = 16;
006079  fsSBDrCrDat = 512;
006080  fsSBDrMdDat = 1024;
006081  fsSBDrBkDat = 2048;
006082  fsSBDrFndrInfo = 4096;
006083  fsSBDrParID = 8192;
006084
006085  { vMAttrib (GetVolParms) bit position constants }
006086  bLimitFCBs = 31;
006087  bLocalWList = 30;
006088  bNoMiniFndr = 29;
006089  bNoVNEdit = 28;
006090  bNoLclSync = 27;
006091  bTrshOffLine = 26;
006092  bNoSwitchTo = 25;
006093  bNoDeskItems = 20;
006094  bNoBootBlks = 19;
006095  bAccessCntl = 18;
006096  bNoSysDir = 17;
006097  bHasExtFSVol = 16;
006098  bHasOpenDeny = 15;
006099  bHasCopyFile = 14;
006100  bHasMoveRename = 13;
006101  bHasDesktopMgr = 12;
006102  bHasShortName = 11;
006103  bHasFolderLock = 10;
006104  bHasPersonalAccessPrivileges = 9;
006105  bHasUserGroupList = 8;
006106  bHasCatSearch = 7;
006107  bHasFileIDs = 6;
```

```
006108  bHasBTreeMgr = 5;
006109  bHasBlankAccessPrivileges = 4;
006110
006111  { Desktop Database icon Constants }
006112  kLargeIcon = 1;
006113  kLarge4BitIcon = 2;
006114  kLarge8BitIcon = 3;
006115  kSmallIcon = 4;
006116  kSmall4BitIcon = 5;
006117  kSmall8BitIcon = 6;
006118
006119  kLargeIconSize = 256;
006120  kLarge4BitIconSize = 512;
006121  kLarge8BitIconSize = 1024;
006122  kSmallIconSize = 64;
006123  kSmall4BitIconSize = 128;
006124  kSmall8BitIconSize = 256;
006125
006126  { Foreign Privilege Model Identifiers }
006127  fsUnixPriv = 1;
006128
006129  { Version Release Stage Codes }
006130  developStage = $20;
006131  alphaStage = $40;
006132  betaStage = $60;
006133  finalStage = $80;
006134
006135  { Authentication Constants }
006136  kNoUserAuthentication = 1;
006137  kPassword = 2;
006138  kEncryptPassword = 3;
006139  kTwoWayEncryptPassword = 6;
006140
006141  TYPE
006142  CInfoType = (hFileInfo,dirInfo);
006143
006144
006145  FXInfo = RECORD
006146   fdIconID: INTEGER;                          {Icon ID}
006147   fdUnused: ARRAY [1..3] OF INTEGER;          {unused but reserved 6 bytes}
006148   fdScript: SignedByte;                       {Script flag and number}
006149   fdXFlags: SignedByte;                       {More flag bits}
006150   fdComment: INTEGER;                         {Comment ID}
006151   fdPutAway: LONGINT;                         {Home Dir ID}
006152   END;
006153
006154  DInfo = RECORD
006155   frRect: Rect;                               {folder rect}
006156   frFlags: INTEGER;                           {Flags}
006157   frLocation: Point;                          {folder location}
006158   frView: INTEGER;                            {folder view}
006159   END;
006160
006161  DXInfo = RECORD
006162   frScroll: Point;                            {scroll position}
006163   frOpenChain: LONGINT;                       {DirID chain of open folders}
```

```
006164   frScript: SignedByte;                    {Script flag and number}
006165   frXFlags: SignedByte;                    {More flag bits}
006166   frComment: INTEGER;                      {comment}
006167   frPutAway: LONGINT;                      {DirID}
006168   END;
006169
006170  GetVolParmsInfoBuffer = RECORD
006171   vMVersion: INTEGER;                      {version number}
006172   vMAttrib: LONGINT;                       {bit vector of attributes (see vMAttrib constants)}
006173   vMLocalHand: Handle;                     {handle to private data}
006174   vMServerAdr: LONGINT;                    {AppleTalk server address or zero}
006175   vMVolumeGrade: LONGINT;                  {approx. speed rating or zero if unrated}
006176   vMForeignPrivID: INTEGER;                {foreign privilege model supported or zero if none}
006177   END;
006178
006179  CInfoPBPtr = ^CInfoPBRec;
006180  CInfoPBRec = RECORD
006181   qLink: QElemPtr;
006182   qType: INTEGER;
006183   ioTrap: INTEGER;
006184   ioCmdAddr: Ptr;
006185   ioCompletion: ProcPtr;
006186   ioResult: OSErr;
006187   ioNamePtr: StringPtr;
006188   ioVRefNum: INTEGER;
006189   ioFRefNum: INTEGER;
006190   ioFVersNum: SignedByte;
006191   filler1: SignedByte;
006192   ioFDirIndex: INTEGER;
006193   ioFlAttrib: SignedByte;
006194   filler2: SignedByte;
006195   CASE CInfoType OF
006196     hFileInfo:
006197    (ioFlFndrInfo: FInfo;
006198    ioDirID: LONGINT;
006199    ioFlStBlk: INTEGER;
006200    ioFlLgLen: LONGINT;
006201    ioFlPyLen: LONGINT;
006202    ioFlRStBlk: INTEGER;
006203    ioFlRLgLen: LONGINT;
006204    ioFlRPyLen: LONGINT;
006205    ioFlCrDat: LONGINT;
006206    ioFlMdDat: LONGINT;
006207    ioFlBkDat: LONGINT;
006208    ioFlXFndrInfo: FXInfo;
006209    ioFlParID: LONGINT;
006210    ioFlClpSiz: LONGINT);
006211     dirInfo:
006212    (ioDrUsrWds: DInfo;
006213    ioDrDirID: LONGINT;
006214    ioDrNmFls: INTEGER;
006215    filler3: ARRAY [1..9] OF INTEGER;
006216    ioDrCrDat: LONGINT;
006217    ioDrMdDat: LONGINT;
006218    ioDrBkDat: LONGINT;
006219    ioDrFndrInfo: DXInfo;
```

```
006220    ioDrParID: LONGINT);
006221   END;
006222
006223  { Catalog position record }
006224  CatPositionRec = RECORD
006225   initialize: LONGINT;
006226   priv: ARRAY [1..6] OF INTEGER;
006227   END;
006228
006229  FSSpecPtr = ^FSSpec;
006230  FSSpecHandle = ^FSSpecPtr;
006231  FSSpec = RECORD
006232   vRefNum: INTEGER;
006233   parID: LONGINT;
006234   name: Str63;
006235   END;
006236
006237  FSSpecArrayPtr = ^FSSpecArray;
006238  FSSpecArrayHandle = ^FSSpecArrayPtr;
006239
006240  FSSpecArray = ARRAY [0..0] OF FSSpec;
006241
006242
006243  { The following are structures to be filled out with the _GetVolMountInfo call
006244   and passed back into the _VolumeMount call for external file system mounts. }
006245
006246  VolumeType = OSType;                        { the "signature" of the file system }
006247
006248  CONST
006249  AppleShareMediaType = 'afpm';               { the signature for AppleShare }
006250
006251  TYPE
006252  VolMountInfoPtr = ^VolMountInfoHeader;
006253  VolMountInfoHeader = RECORD
006254   length: INTEGER;                           { length of location data (including self) }
006255   media: VolumeType;                         { type of media.  Variable length data follows }
006256   END;
006257
006258  AFPVolMountInfoPtr = ^AFPVolMountInfo;
006259  AFPVolMountInfo = RECORD
006260   length: INTEGER;                           { length of location data (including self) }
006261   media: VolumeType;                         { type of media }
006262   flags: INTEGER;                            { bits for no messages, no reconnect }
006263   nbpInterval: SignedByte;                   { NBP Interval parameter (IM2, p.322) }
006264   nbpCount: SignedByte;                      { NBP Interval parameter (IM2, p.322) }
006265   uamType: INTEGER;                          { User Authentication Method }
006266   zoneNameOffset: INTEGER;                   { short positive offset from start of struct to Zone Name }
006267   serverNameOffset: INTEGER;                 { offset to pascal Server Name string }
006268   volNameOffset: INTEGER;                    { offset to pascal Volume Name string }
006269   userNameOffset: INTEGER;                   { offset to pascal User Name string }
006270   userPasswordOffset: INTEGER;               { offset to pascal User Password string }
006271   volPasswordOffset: INTEGER;                { offset to pascal Volume Password string }
006272   AFPData: PACKED ARRAY [1..144] OF CHAR;    { variable length data may follow }
006273   END;
006274
006275  DTPBPtr = ^DTPBRec;
```

```
006276  DTPBRec = RECORD
006277   qLink: QElemPtr;
006278   qType: INTEGER;
006279   ioTrap: INTEGER;
006280   ioCmdAddr: Ptr;
006281   ioCompletion: ProcPtr;
006282   ioResult: OSErr;
006283   ioNamePtr: StringPtr;
006284   ioVRefNum: INTEGER;
006285   ioDTRefNum: INTEGER;
006286   ioIndex: INTEGER;
006287   ioTagInfo: LONGINT;
006288   ioDTBuffer: Ptr;
006289   ioDTReqCount: LONGINT;
006290   ioDTActCount: LONGINT;
006291   filler1: SignedByte;
006292   ioIconType: SignedByte;
006293   filler2: INTEGER;
006294   ioDirID: LONGINT;
006295   ioFileCreator: OSType;
006296   ioFileType: OSType;
006297   ioFiller3: LONGINT;
006298   ioDTLgLen: LONGINT;
006299   ioDTPyLen: LONGINT;
006300   ioFiller4: ARRAY [1..14] OF INTEGER;
006301   ioAPPLParID: LONGINT;
006302   END;
006303
006304  HParmBlkPtr = ^HParamBlockRec;
006305  HParamBlockRec = RECORD
006306   qLink: QElemPtr;
006307   qType: INTEGER;
006308   ioTrap: INTEGER;
006309   ioCmdAddr: Ptr;
006310   ioCompletion: ProcPtr;
006311   ioResult: OSErr;
006312   ioNamePtr: StringPtr;
006313   ioVRefNum: INTEGER;
006314   CASE ParamBlkType OF
006315     IOParam:
006316    (ioRefNum: INTEGER;
006317    ioVersNum: SignedByte;
006318    ioPermssn: SignedByte;
006319    ioMisc: Ptr;
006320    ioBuffer: Ptr;
006321    ioReqCount: LONGINT;                    {size of buffer area}
006322    ioActCount: LONGINT;                    {length of vol parms data}
006323    ioPosMode: INTEGER;
006324    ioPosOffset: LONGINT);
006325     FileParam:
006326    (ioFRefNum: INTEGER;
006327    ioFVersNum: SignedByte;
006328    filler1: SignedByte;
006329    ioFDirIndex: INTEGER;
006330    ioFlAttrib: SignedByte;
006331    ioFlVersNum: SignedByte;
```

```
006332    ioFlFndrInfo: FInfo;
006333    ioDirID: LONGINT;
006334    ioFlStBlk: INTEGER;
006335    ioFlLgLen: LONGINT;
006336    ioFlPyLen: LONGINT;
006337    ioFlRStBlk: INTEGER;
006338    ioFlRLgLen: LONGINT;
006339    ioFlRPyLen: LONGINT;
006340    ioFlCrDat: LONGINT;
006341    ioFlMdDat: LONGINT);
006342     VolumeParam:
006343    (filler2: LONGINT;
006344    ioVolIndex: INTEGER;
006345    ioVCrDate: LONGINT;
006346    ioVLsMod: LONGINT;
006347    ioVAtrb: INTEGER;
006348    ioVNmFls: INTEGER;
006349    ioVBitMap: INTEGER;
006350    ioAllocPtr: INTEGER;
006351    ioVNmAlBlks: INTEGER;
006352    ioVAlBlkSiz: LONGINT;
006353    ioVClpSiz: LONGINT;
006354    ioAlBlSt: INTEGER;
006355    ioVNxtCNID: LONGINT;
006356    ioVFrBlk: INTEGER;
006357    ioVSigWord: INTEGER;
006358    ioVDrvInfo: INTEGER;
006359    ioVDRefNum: INTEGER;
006360    ioVFSID: INTEGER;
006361    ioVBkUp: LONGINT;
006362    ioVSeqNum: INTEGER;
006363    ioVWrCnt: LONGINT;
006364    ioVFilCnt: LONGINT;
006365    ioVDirCnt: LONGINT;
006366    ioVFndrInfo: ARRAY [1..8] OF LONGINT);
006367     AccessParam:
006368    (filler3: INTEGER;
006369    ioDenyModes: INTEGER;                    {access rights data}
006370    filler4: INTEGER;
006371    filler5: SignedByte;
006372    ioACUser: SignedByte;                    {access rights for directory only}
006373    filler6: LONGINT;
006374    ioACOwnerID: LONGINT;                    {owner ID}
006375    ioACGroupID: LONGINT;                    {group ID}
006376    ioACAccess: LONGINT);                    {access rights}
006377     ObjParam:
006378    (filler7: INTEGER;
006379    ioObjType: INTEGER;                      {function code}
006380    ioObjNamePtr: Ptr;                       {ptr to returned creator/group name}
006381    ioObjID: LONGINT);                       {creator/group ID}
006382     CopyParam:
006383    (ioDstVRefNum: INTEGER;                  {destination vol identifier}
006384    filler8: INTEGER;
006385    ioNewName: Ptr;                          {ptr to destination pathname}
006386    ioCopyName: Ptr;                         {ptr to optional name}
006387    ioNewDirID: LONGINT);                    {destination directory ID}
```

```
006388     WDParam:
006389    (filler9: INTEGER;
006390    ioWDIndex: INTEGER;
006391    ioWDProcID: LONGINT;
006392    ioWDVRefNum: INTEGER;
006393    filler10: INTEGER;
006394    filler11: LONGINT;
006395    filler12: LONGINT;
006396    filler13: LONGINT;
006397    ioWDDirID: LONGINT);
006398     FIDParam:
006399    (filler14: LONGINT;
006400    ioDestNamePtr: StringPtr;
006401    filler15: LONGINT;
006402    ioDestDirID: LONGINT;
006403    filler16: LONGINT;
006404    filler17: LONGINT;
006405    ioSrcDirID: LONGINT;
006406    filler18:  INTEGER;
006407    ioFileID:  LONGINT);
006408     CSParam:
006409    (ioMatchPtr: FSSpecArrayPtr;              {match array}
006410    ioReqMatchCount: LONGINT;                 {maximum allowable matches}
006411    ioActMatchCount: LONGINT;                 {actual match count}
006412    ioSearchBits: LONGINT;                    {search criteria selector}
006413    ioSearchInfo1: CInfoPBPtr;                {search values and range lower bounds}
006414    ioSearchInfo2: CInfoPBPtr;                {search values and range upper bounds}
006415    ioSearchTime: LONGINT;                    {length of time to run the search}
006416    ioCatPosition: CatPositionRec;            {current position in the catalog}
006417    ioOptBuffer: Ptr;                         {optional performance enhancement buffer}
006418    ioOptBufSize: LONGINT);                   {length of buffer pointed to by ioOptBuffer}
006419     ForeignPrivParam:
006420      (filler21: LONGINT;
006421    filler22: LONGINT;
006422    ioForeignPrivBuffer: Ptr;
006423    ioForeignPrivReqCount: LONGINT;
006424    ioForeignPrivActCount: LONGINT;
006425    filler23: LONGINT;
006426    ioForeignPrivDirID: LONGINT;
006427    ioForeignPrivInfo1: LONGINT;
006428    ioForeignPrivInfo2: LONGINT;
006429    ioForeignPrivInfo3: LONGINT;
006430    ioForeignPrivInfo4: LONGINT);
006431   END;
006432
006433  CMovePBPtr = ^CMovePBRec;
006434  CMovePBRec = RECORD
006435   qLink: QElemPtr;
006436   qType: INTEGER;
006437   ioTrap: INTEGER;
006438   ioCmdAddr: Ptr;
006439   ioCompletion: ProcPtr;
006440   ioResult: OSErr;
006441   ioNamePtr: StringPtr;
006442   ioVRefNum: INTEGER;
006443   filler1: LONGINT;
```

```
006444   ioNewName: StringPtr;
006445   filler2: LONGINT;
006446   ioNewDirID: LONGINT;
006447   filler3: ARRAY [1..2] OF LONGINT;
006448   ioDirID: LONGINT;
006449   END;
006450
006451  WDPBPtr = ^WDPBRec;
006452  WDPBRec = RECORD
006453   qLink: QElemPtr;
006454   qType: INTEGER;
006455   ioTrap: INTEGER;
006456   ioCmdAddr: Ptr;
006457   ioCompletion: ProcPtr;
006458   ioResult: OSErr;
006459   ioNamePtr: StringPtr;
006460   ioVRefNum: INTEGER;
006461   filler1: INTEGER;
006462   ioWDIndex: INTEGER;
006463   ioWDProcID: LONGINT;
006464   ioWDVRefNum: INTEGER;
006465   filler2: ARRAY [1..7] OF INTEGER;
006466   ioWDDirID: LONGINT;
006467   END;
006468
006469  FCBPBPtr = ^FCBPBRec;
006470  FCBPBRec = RECORD
006471   qLink: QElemPtr;
006472   qType: INTEGER;
006473   ioTrap: INTEGER;
006474   ioCmdAddr: Ptr;
006475   ioCompletion: ProcPtr;
006476   ioResult: OSErr;
006477   ioNamePtr: StringPtr;
006478   ioVRefNum: INTEGER;
006479   ioRefNum: INTEGER;
006480   filler: INTEGER;
006481   ioFCBIndx: INTEGER;
006482   filler1: INTEGER;
006483   ioFCBFlNm: LONGINT;
006484   ioFCBFlags: INTEGER;
006485   ioFCBStBlk: INTEGER;
006486   ioFCBEOF: LONGINT;
006487   ioFCBPLen: LONGINT;
006488   ioFCBCrPs: LONGINT;
006489   ioFCBVRefNum: INTEGER;
006490   ioFCBClpSiz: LONGINT;
006491   ioFCBParID: LONGINT;
006492   END;
006493
006494  { Numeric version part of 'vers' resource }
006495  NumVersion = PACKED RECORD
006496   CASE INTEGER OF
006497     0:
006498    (majorRev: SignedByte;                    {1st part of version number in BCD}
006499     minorRev: 0..9;                          {2nd part is 1 nibble in BCD}
```

```
006500    bugFixRev: 0..9;                        {3rd part is 1 nibble in BCD}
006501    stage: SignedByte;                      {stage code: dev, alpha, beta, final}
006502    nonRelRev: SignedByte);                 {revision level of non-released version}
006503     1:
006504    (version: LONGINT);                     {to use all 4 fields at one time}
006505   END;
006506
006507  { 'vers' resource format }
006508  VersRecPtr = ^VersRec;
006509  VersRecHndl = ^VersRecPtr;
006510  VersRec = RECORD
006511   numericVersion: NumVersion;               {encoded version number}
006512   countryCode: INTEGER;                     {country code from intl utilities}
006513   shortVersion: Str255;                     {version number string - worst case}
006514   reserved: Str255;                         {longMessage string packed after shortVersion}
006515   END;
006516
006517
006518  FUNCTION PBOpen(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006519  FUNCTION PBOpenSync(paramBlock: ParmBlkPtr): OSErr;
006520   INLINE $205F,$A000,$3E80;
006521  FUNCTION PBOpenAsync(paramBlock: ParmBlkPtr): OSErr;
006522   INLINE $205F,$A400,$3E80;
006523  FUNCTION PBClose(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006524  FUNCTION PBCloseSync(paramBlock: ParmBlkPtr): OSErr;
006525   INLINE $205F,$A001,$3E80;
006526  FUNCTION PBCloseAsync(paramBlock: ParmBlkPtr): OSErr;
006527   INLINE $205F,$A401,$3E80;
006528  FUNCTION PBRead(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006529  FUNCTION PBReadSync(paramBlock: ParmBlkPtr): OSErr;
006530   INLINE $205F,$A002,$3E80;
006531  FUNCTION PBReadAsync(paramBlock: ParmBlkPtr): OSErr;
006532   INLINE $205F,$A402,$3E80;
006533  FUNCTION PBWrite(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006534  FUNCTION PBWriteSync(paramBlock: ParmBlkPtr): OSErr;
006535   INLINE $205F,$A003,$3E80;
006536  FUNCTION PBWriteAsync(paramBlock: ParmBlkPtr): OSErr;
006537   INLINE $205F,$A403,$3E80;
006538  FUNCTION PBGetVInfo(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006539  FUNCTION PBGetVInfoSync(paramBlock: ParmBlkPtr): OSErr;
006540   INLINE $205F,$A007,$3E80;
006541  FUNCTION PBGetVInfoAsync(paramBlock: ParmBlkPtr): OSErr;
006542   INLINE $205F,$A407,$3E80;
006543  FUNCTION PBGetVol(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006544  FUNCTION PBGetVolSync(paramBlock: ParmBlkPtr): OSErr;
006545   INLINE $205F,$A014,$3E80;
006546  FUNCTION PBGetVolAsync(paramBlock: ParmBlkPtr): OSErr;
006547   INLINE $205F,$A414,$3E80;
006548  FUNCTION PBSetVol(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006549  FUNCTION PBSetVolSync(paramBlock: ParmBlkPtr): OSErr;
006550   INLINE $205F,$A015,$3E80;
006551  FUNCTION PBSetVolAsync(paramBlock: ParmBlkPtr): OSErr;
006552   INLINE $205F,$A415,$3E80;
006553  FUNCTION PBFlushVol(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006554  FUNCTION PBFlushVolSync(paramBlock: ParmBlkPtr): OSErr;
006555   INLINE $205F,$A013,$3E80;
```

```
006556  FUNCTION PBFlushVolAsync(paramBlock: ParmBlkPtr): OSErr;
006557   INLINE $205F,$A413,$3E80;
006558  FUNCTION PBCreate(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006559  FUNCTION PBCreateSync(paramBlock: ParmBlkPtr): OSErr;
006560   INLINE $205F,$A008,$3E80;
006561  FUNCTION PBCreateAsync(paramBlock: ParmBlkPtr): OSErr;
006562   INLINE $205F,$A408,$3E80;
006563  FUNCTION PBDelete(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006564  FUNCTION PBDeleteSync(paramBlock: ParmBlkPtr): OSErr;
006565   INLINE $205F,$A009,$3E80;
006566  FUNCTION PBDeleteAsync(paramBlock: ParmBlkPtr): OSErr;
006567   INLINE $205F,$A409,$3E80;
006568  FUNCTION PBOpenDF(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006569  FUNCTION PBOpenDFSync(paramBlock: ParmBlkPtr): OSErr;
006570   INLINE $205F,$701A,$A060,$3E80;
006571  FUNCTION PBOpenDFAsync(paramBlock: ParmBlkPtr): OSErr;
006572   INLINE $205F,$701A,$A460,$3E80;
006573  FUNCTION PBOpenRF(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006574  FUNCTION PBOpenRFSync(paramBlock: ParmBlkPtr): OSErr;
006575   INLINE $205F,$A00A,$3E80;
006576  FUNCTION PBOpenRFAsync(paramBlock: ParmBlkPtr): OSErr;
006577   INLINE $205F,$A40A,$3E80;
006578  FUNCTION PBRename(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006579  FUNCTION PBRenameSync(paramBlock: ParmBlkPtr): OSErr;
006580   INLINE $205F,$A00B,$3E80;
006581  FUNCTION PBRenameAsync(paramBlock: ParmBlkPtr): OSErr;
006582   INLINE $205F,$A40B,$3E80;
006583  FUNCTION PBGetFInfo(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006584  FUNCTION PBGetFInfoSync(paramBlock: ParmBlkPtr): OSErr;
006585   INLINE $205F,$A00C,$3E80;
006586  FUNCTION PBGetFInfoAsync(paramBlock: ParmBlkPtr): OSErr;
006587   INLINE $205F,$A40C,$3E80;
006588  FUNCTION PBSetFInfo(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006589  FUNCTION PBSetFInfoSync(paramBlock: ParmBlkPtr): OSErr;
006590   INLINE $205F,$A00D,$3E80;
006591  FUNCTION PBSetFInfoAsync(paramBlock: ParmBlkPtr): OSErr;
006592   INLINE $205F,$A40D,$3E80;
006593  FUNCTION PBSetFLock(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006594  FUNCTION PBSetFLockSync(paramBlock: ParmBlkPtr): OSErr;
006595   INLINE $205F,$A041,$3E80;
006596  FUNCTION PBSetFLockAsync(paramBlock: ParmBlkPtr): OSErr;
006597   INLINE $205F,$A441,$3E80;
006598  FUNCTION PBRstFLock(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006599  FUNCTION PBRstFLockSync(paramBlock: ParmBlkPtr): OSErr;
006600   INLINE $205F,$A042,$3E80;
006601  FUNCTION PBRstFLockAsync(paramBlock: ParmBlkPtr): OSErr;
006602   INLINE $205F,$A442,$3E80;
006603  FUNCTION PBSetFVers(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006604  FUNCTION PBSetFVersSync(paramBlock: ParmBlkPtr): OSErr;
006605   INLINE $205F,$A043,$3E80;
006606  FUNCTION PBSetFVersAsync(paramBlock: ParmBlkPtr): OSErr;
006607   INLINE $205F,$A443,$3E80;
006608  FUNCTION PBAllocate(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006609  FUNCTION PBAllocateSync(paramBlock: ParmBlkPtr): OSErr;
006610   INLINE $205F,$A010,$3E80;
006611  FUNCTION PBAllocateAsync(paramBlock: ParmBlkPtr): OSErr;
```

```
006612   INLINE $205F,$A410,$3E80;
006613  FUNCTION PBGetEOF(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006614  FUNCTION PBGetEOFSync(paramBlock: ParmBlkPtr): OSErr;
006615   INLINE $205F,$A011,$3E80;
006616  FUNCTION PBGetEOFAsync(paramBlock: ParmBlkPtr): OSErr;
006617   INLINE $205F,$A411,$3E80;
006618  FUNCTION PBSetEOF(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006619  FUNCTION PBSetEOFSync(paramBlock: ParmBlkPtr): OSErr;
006620   INLINE $205F,$A012,$3E80;
006621  FUNCTION PBSetEOFAsync(paramBlock: ParmBlkPtr): OSErr;
006622   INLINE $205F,$A412,$3E80;
006623  FUNCTION PBGetFPos(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006624  FUNCTION PBGetFPosSync(paramBlock: ParmBlkPtr): OSErr;
006625   INLINE $205F,$A018,$3E80;
006626  FUNCTION PBGetFPosAsync(paramBlock: ParmBlkPtr): OSErr;
006627   INLINE $205F,$A418,$3E80;
006628  FUNCTION PBSetFPos(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006629  FUNCTION PBSetFPosSync(paramBlock: ParmBlkPtr): OSErr;
006630   INLINE $205F,$A044,$3E80;
006631  FUNCTION PBSetFPosAsync(paramBlock: ParmBlkPtr): OSErr;
006632   INLINE $205F,$A444,$3E80;
006633  FUNCTION PBFlushFile(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006634  FUNCTION PBFlushFileSync(paramBlock: ParmBlkPtr): OSErr;
006635   INLINE $205F,$A045,$3E80;
006636  FUNCTION PBFlushFileAsync(paramBlock: ParmBlkPtr): OSErr;
006637   INLINE $205F,$A445,$3E80;
006638  FUNCTION PBMountVol(paramBlock: ParmBlkPtr): OSErr;
006639  FUNCTION PBUnmountVol(paramBlock: ParmBlkPtr): OSErr;
006640  FUNCTION PBEject(paramBlock: ParmBlkPtr): OSErr;
006641  FUNCTION PBOffLine(paramBlock: ParmBlkPtr): OSErr;
006642
006643  FUNCTION PBCatSearch(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006644  FUNCTION PBCatSearchSync(paramBlock: HParmBlkPtr): OSErr;
006645   INLINE $205F,$7018,$A260,$3E80;
006646  FUNCTION PBCatSearchAsync(paramBlock: HParmBlkPtr): OSErr;
006647   INLINE $205F,$7018,$A660,$3E80;
006648
006649  PROCEDURE AddDrive(drvrRefNum: INTEGER;drvNum: INTEGER;qEl: DrvQElPtr);
006650  FUNCTION FSOpen(fileName: Str255;vRefNum: INTEGER;VAR refNum: INTEGER): OSErr;
006651  FUNCTION OpenDF(fileName: Str255;vRefNum: INTEGER;VAR refNum: INTEGER): OSErr;
006652  FUNCTION FSClose(refNum: INTEGER): OSErr;
006653  FUNCTION FSRead(refNum: INTEGER;VAR count: LONGINT;buffPtr: Ptr): OSErr;
006654  FUNCTION FSWrite(refNum: INTEGER;VAR count: LONGINT;buffPtr: Ptr): OSErr;
006655  FUNCTION GetVInfo(drvNum: INTEGER;volName: StringPtr;VAR vRefNum: INTEGER;
006656   VAR freeBytes: LONGINT): OSErr;
006657  FUNCTION GetFInfo(fileName: Str255;vRefNum: INTEGER;VAR fndrInfo: FInfo): OSErr;
006658  FUNCTION GetVol(volName: StringPtr;VAR vRefNum: INTEGER): OSErr;
006659  FUNCTION SetVol(volName: StringPtr;vRefNum: INTEGER): OSErr;
006660  FUNCTION UnmountVol(volName: StringPtr;vRefNum: INTEGER): OSErr;
006661  FUNCTION Eject(volName: StringPtr;vRefNum: INTEGER): OSErr;
006662  FUNCTION FlushVol(volName: StringPtr;vRefNum: INTEGER): OSErr;
006663  FUNCTION Create(fileName: Str255;vRefNum: INTEGER;creator: OSType;fileType: OSType): OSErr;
006664  FUNCTION FSDelete(fileName: Str255;vRefNum: INTEGER): OSErr;
006665  FUNCTION OpenRF(fileName: Str255;vRefNum: INTEGER;VAR refNum: INTEGER): OSErr;
006666  FUNCTION Rename(oldName: Str255;vRefNum: INTEGER;newName: Str255): OSErr;
006667  FUNCTION SetFInfo(fileName: Str255;vRefNum: INTEGER;fndrInfo: FInfo): OSErr;
```

```
006668  FUNCTION SetFLock(fileName: Str255;vRefNum: INTEGER): OSErr;
006669  FUNCTION RstFLock(fileName: Str255;vRefNum: INTEGER): OSErr;
006670  FUNCTION Allocate(refNum: INTEGER;VAR count: LONGINT): OSErr;
006671  FUNCTION GetEOF(refNum: INTEGER;VAR logEOF: LONGINT): OSErr;
006672  FUNCTION SetEOF(refNum: INTEGER;logEOF: LONGINT): OSErr;
006673  FUNCTION GetFPos(refNum: INTEGER;VAR filePos: LONGINT): OSErr;
006674  FUNCTION SetFPos(refNum: INTEGER;posMode: INTEGER;posOff: LONGINT): OSErr;
006675  FUNCTION GetVRefNum(fileRefNum: INTEGER;VAR vRefNum: INTEGER): OSErr;
006676
006677  FUNCTION PBOpenWD(paramBlock: WDPBPtr;async: BOOLEAN): OSErr;
006678  FUNCTION PBOpenWDSync(paramBlock: WDPBPtr): OSErr;
006679   INLINE $205F,$7001,$A260,$3E80;
006680  FUNCTION PBOpenWDAsync(paramBlock: WDPBPtr): OSErr;
006681   INLINE $205F,$7001,$A660,$3E80;
006682  FUNCTION PBCloseWD(paramBlock: WDPBPtr;async: BOOLEAN): OSErr;
006683  FUNCTION PBCloseWDSync(paramBlock: WDPBPtr): OSErr;
006684   INLINE $205F,$7002,$A260,$3E80;
006685  FUNCTION PBCloseWDAsync(paramBlock: WDPBPtr): OSErr;
006686   INLINE $205F,$7002,$A660,$3E80;
006687  FUNCTION PBHSetVol(paramBlock: WDPBPtr;async: BOOLEAN): OSErr;
006688  FUNCTION PBHSetVolSync(paramBlock: WDPBPtr): OSErr;
006689   INLINE $205F,$A215,$3E80;
006690  FUNCTION PBHSetVolAsync(paramBlock: WDPBPtr): OSErr;
006691   INLINE $205F,$A615,$3E80;
006692  FUNCTION PBHGetVol(paramBlock: WDPBPtr;async: BOOLEAN): OSErr;
006693  FUNCTION PBHGetVolSync(paramBlock: WDPBPtr): OSErr;
006694   INLINE $205F,$A214,$3E80;
006695  FUNCTION PBHGetVolAsync(paramBlock: WDPBPtr): OSErr;
006696   INLINE $205F,$A614,$3E80;
006697  FUNCTION PBCatMove(paramBlock: CMovePBPtr;async: BOOLEAN): OSErr;
006698  FUNCTION PBCatMoveSync(paramBlock: CMovePBPtr): OSErr;
006699   INLINE $205F,$7005,$A260,$3E80;
006700  FUNCTION PBCatMoveAsync(paramBlock: CMovePBPtr): OSErr;
006701   INLINE $205F,$7005,$A660,$3E80;
006702  FUNCTION PBDirCreate(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006703  FUNCTION PBDirCreateSync(paramBlock: HParmBlkPtr): OSErr;
006704   INLINE $205F,$7006,$A260,$3E80;
006705  FUNCTION PBDirCreateAsync(paramBlock: HParmBlkPtr): OSErr;
006706   INLINE $205F,$7006,$A660,$3E80;
006707  FUNCTION PBGetWDInfo(paramBlock: WDPBPtr;async: BOOLEAN): OSErr;
006708  FUNCTION PBGetWDInfoSync(paramBlock: WDPBPtr): OSErr;
006709   INLINE $205F,$7007,$A260,$3E80;
006710  FUNCTION PBGetWDInfoAsync(paramBlock: WDPBPtr): OSErr;
006711   INLINE $205F,$7007,$A660,$3E80;
006712  FUNCTION PBGetFCBInfo(paramBlock: FCBPBPtr;async: BOOLEAN): OSErr;
006713  FUNCTION PBGetFCBInfoSync(paramBlock: FCBPBPtr): OSErr;
006714   INLINE $205F,$7008,$A260,$3E80;
006715  FUNCTION PBGetFCBInfoAsync(paramBlock: FCBPBPtr): OSErr;
006716   INLINE $205F,$7008,$A660,$3E80;
006717  FUNCTION PBGetCatInfo(paramBlock: CInfoPBPtr;async: BOOLEAN): OSErr;
006718  FUNCTION PBGetCatInfoSync(paramBlock: CInfoPBPtr): OSErr;
006719   INLINE $205F,$7009,$A260,$3E80;
006720  FUNCTION PBGetCatInfoAsync(paramBlock: CInfoPBPtr): OSErr;
006721   INLINE $205F,$7009,$A660,$3E80;
006722  FUNCTION PBSetCatInfo(paramBlock: CInfoPBPtr;async: BOOLEAN): OSErr;
006723  FUNCTION PBSetCatInfoSync(paramBlock: CInfoPBPtr): OSErr;
```

```
006724   INLINE $205F,$700A,$A260,$3E80;
006725  FUNCTION PBSetCatInfoAsync(paramBlock: CInfoPBPtr): OSErr;
006726   INLINE $205F,$700A,$A660,$3E80;
006727
006728  FUNCTION PBAllocContig(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006729  FUNCTION PBAllocContigSync(paramBlock: ParmBlkPtr): OSErr;
006730   INLINE $205F,$A210,$3E80;
006731  FUNCTION PBAllocContigAsync(paramBlock: ParmBlkPtr): OSErr;
006732   INLINE $205F,$A610,$3E80;
006733  FUNCTION PBLockRange(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006734  FUNCTION PBLockRangeSync(paramBlock: ParmBlkPtr): OSErr;
006735   INLINE $205F,$7010,$A260,$3E80;
006736  FUNCTION PBLockRangeAsync(paramBlock: ParmBlkPtr): OSErr;
006737   INLINE $205F,$7010,$A660,$3E80;
006738  FUNCTION PBUnlockRange(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
006739  FUNCTION PBUnlockRangeSync(paramBlock: ParmBlkPtr): OSErr;
006740   INLINE $205F,$7011,$A260,$3E80;
006741  FUNCTION PBUnlockRangeAsync(paramBlock: ParmBlkPtr): OSErr;
006742   INLINE $205F,$7011,$A660,$3E80;
006743  FUNCTION PBSetVInfo(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006744  FUNCTION PBSetVInfoSync(paramBlock: HParmBlkPtr): OSErr;
006745   INLINE $205F,$700B,$A260,$3E80;
006746  FUNCTION PBSetVInfoAsync(paramBlock: HParmBlkPtr): OSErr;
006747   INLINE $205F,$700B,$A660,$3E80;
006748  FUNCTION PBHGetVInfo(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006749  FUNCTION PBHGetVInfoSync(paramBlock: HParmBlkPtr): OSErr;
006750   INLINE $205F,$A207,$3E80;
006751  FUNCTION PBHGetVInfoAsync(paramBlock: HParmBlkPtr): OSErr;
006752   INLINE $205F,$A607,$3E80;
006753  FUNCTION PBHOpen(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006754  FUNCTION PBHOpenSync(paramBlock: HParmBlkPtr): OSErr;
006755   INLINE $205F,$A200,$3E80;
006756  FUNCTION PBHOpenAsync(paramBlock: HParmBlkPtr): OSErr;
006757   INLINE $205F,$A600,$3E80;
006758  FUNCTION PBHOpenRF(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006759  FUNCTION PBHOpenRFSync(paramBlock: HParmBlkPtr): OSErr;
006760   INLINE $205F,$A20A,$3E80;
006761  FUNCTION PBHOpenRFAsync(paramBlock: HParmBlkPtr): OSErr;
006762   INLINE $205F,$A60A,$3E80;
006763  FUNCTION PBHOpenDF(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006764  FUNCTION PBHOpenDFSync(paramBlock: HParmBlkPtr): OSErr;
006765   INLINE $205F,$701A,$A260,$3E80;
006766  FUNCTION PBHOpenDFAsync(paramBlock: HParmBlkPtr): OSErr;
006767   INLINE $205F,$701A,$A660,$3E80;
006768
006769  FUNCTION PBHCreate(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006770  FUNCTION PBHCreateSync(paramBlock: HParmBlkPtr): OSErr;
006771   INLINE $205F,$A208,$3E80;
006772  FUNCTION PBHCreateAsync(paramBlock: HParmBlkPtr): OSErr;
006773   INLINE $205F,$A608,$3E80;
006774  FUNCTION PBHDelete(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006775  FUNCTION PBHDeleteSync(paramBlock: HParmBlkPtr): OSErr;
006776   INLINE $205F,$A209,$3E80;
006777  FUNCTION PBHDeleteAsync(paramBlock: HParmBlkPtr): OSErr;
006778   INLINE $205F,$A609,$3E80;
006779  FUNCTION PBHRename(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
```

```
006780  FUNCTION PBHRenameSync(paramBlock: HParmBlkPtr): OSErr;
006781   INLINE $205F,$A20B,$3E80;
006782  FUNCTION PBHRenameAsync(paramBlock: HParmBlkPtr): OSErr;
006783   INLINE $205F,$A60B,$3E80;
006784  FUNCTION PBHRstFLock(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006785  FUNCTION PBHRstFLockSync(paramBlock: HParmBlkPtr): OSErr;
006786   INLINE $205F,$A242,$3E80;
006787  FUNCTION PBHRstFLockAsync(paramBlock: HParmBlkPtr): OSErr;
006788   INLINE $205F,$A642,$3E80;
006789  FUNCTION PBHSetFLock(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006790  FUNCTION PBHSetFLockSync(paramBlock: HParmBlkPtr): OSErr;
006791   INLINE $205F,$A241,$3E80;
006792  FUNCTION PBHSetFLockAsync(paramBlock: HParmBlkPtr): OSErr;
006793   INLINE $205F,$A641,$3E80;
006794  FUNCTION PBHGetFInfo(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006795  FUNCTION PBHGetFInfoSync(paramBlock: HParmBlkPtr): OSErr;
006796   INLINE $205F,$A20C,$3E80;
006797  FUNCTION PBHGetFInfoAsync(paramBlock: HParmBlkPtr): OSErr;
006798   INLINE $205F,$A60C,$3E80;
006799  FUNCTION PBHSetFInfo(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006800  FUNCTION PBHSetFInfoSync(paramBlock: HParmBlkPtr): OSErr;
006801   INLINE $205F,$A20D,$3E80;
006802  FUNCTION PBHSetFInfoAsync(paramBlock: HParmBlkPtr): OSErr;
006803   INLINE $205F,$A60D,$3E80;
006804
006805  FUNCTION PBMakeFSSpec(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006806  FUNCTION PBMakeFSSpecSync(paramBlock: HParmBlkPtr): OSErr;
006807   INLINE $205F,$701B,$A260,$3E80;
006808  FUNCTION PBMakeFSSpecAsync(paramBlock: HParmBlkPtr): OSErr;
006809   INLINE $205F,$701B,$A660,$3E80;
006810
006811  PROCEDURE FInitQueue;
006812   INLINE $A016;
006813  FUNCTION GetFSQHdr: QHdrPtr;
006814   INLINE $2EBC,$0000,$0360;
006815  FUNCTION GetDrvQHdr: QHdrPtr;
006816   INLINE $2EBC,$0000,$0308;
006817  FUNCTION GetVCBQHdr: QHdrPtr;
006818   INLINE $2EBC,$0000,$0356;
006819  FUNCTION HGetVol(volName: StringPtr;VAR vRefNum: INTEGER;VAR dirID: LONGINT): OSErr;
006820  FUNCTION HSetVol(volName: StringPtr;vRefNum: INTEGER;dirID: LONGINT): OSErr;
006821  FUNCTION HOpen(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;permission: SignedByte;
006822   VAR refNum: INTEGER): OSErr;
006823  FUNCTION HOpenDF(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;permission: SignedByte;
006824   VAR refNum: INTEGER): OSErr;
006825  FUNCTION HOpenRF(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;permission: SignedByte;
006826   VAR refNum: INTEGER): OSErr;
006827  FUNCTION AllocContig(refNum: INTEGER;VAR count: LONGINT): OSErr;
006828  FUNCTION HCreate(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;creator: OSType;
006829   fileType: OSType): OSErr;
006830  FUNCTION DirCreate(vRefNum: INTEGER;parentDirID: LONGINT;directoryName: Str255;
006831   VAR createdDirID: LONGINT): OSErr;
006832  FUNCTION HDelete(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255): OSErr;
006833  FUNCTION HGetFInfo(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;VAR fndrInfo: FInfo): OSErr;
006834  FUNCTION HSetFInfo(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;fndrInfo: FInfo): OSErr;
006835  FUNCTION HSetFLock(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255): OSErr;
```

```
006836  FUNCTION HRstFLock(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255): OSErr;
006837  FUNCTION HRename(vRefNum: INTEGER;dirID: LONGINT;oldName: Str255;newName: Str255): OSErr;
006838  FUNCTION CatMove(vRefNum: INTEGER;dirID: LONGINT;oldName: Str255;newDirID: LONGINT;
006839   newName: Str255): OSErr;
006840  FUNCTION OpenWD(vRefNum: INTEGER;dirID: LONGINT;procID: LONGINT;VAR wdRefNum: INTEGER): OSErr;
006841  FUNCTION CloseWD(wdRefNum: INTEGER): OSErr;
006842  FUNCTION GetWDInfo(wdRefNum: INTEGER;VAR vRefNum: INTEGER;VAR dirID: LONGINT;
006843   VAR procID: LONGINT): OSErr;
006844
006845  {  shared environment  }
006846  FUNCTION PBHGetVolParms(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006847  FUNCTION PBHGetVolParmsSync(paramBlock: HParmBlkPtr): OSErr;
006848   INLINE $205F,$7030,$A260,$3E80;
006849  FUNCTION PBHGetVolParmsAsync(paramBlock: HParmBlkPtr): OSErr;
006850   INLINE $205F,$7030,$A660,$3E80;
006851  FUNCTION PBHGetLogInInfo(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006852  FUNCTION PBHGetLogInInfoSync(paramBlock: HParmBlkPtr): OSErr;
006853   INLINE $205F,$7031,$A260,$3E80;
006854  FUNCTION PBHGetLogInInfoAsync(paramBlock: HParmBlkPtr): OSErr;
006855   INLINE $205F,$7031,$A660,$3E80;
006856  FUNCTION PBHGetDirAccess(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006857  FUNCTION PBHGetDirAccessSync(paramBlock: HParmBlkPtr): OSErr;
006858   INLINE $205F,$7032,$A260,$3E80;
006859  FUNCTION PBHGetDirAccessAsync(paramBlock: HParmBlkPtr): OSErr;
006860   INLINE $205F,$7032,$A660,$3E80;
006861  FUNCTION PBHSetDirAccess(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006862  FUNCTION PBHSetDirAccessSync(paramBlock: HParmBlkPtr): OSErr;
006863   INLINE $205F,$7033,$A260,$3E80;
006864  FUNCTION PBHSetDirAccessAsync(paramBlock: HParmBlkPtr): OSErr;
006865   INLINE $205F,$7033,$A660,$3E80;
006866  FUNCTION PBHMapID(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006867  FUNCTION PBHMapIDSync(paramBlock: HParmBlkPtr): OSErr;
006868   INLINE $205F,$7034,$A260,$3E80;
006869  FUNCTION PBHMapIDAsync(paramBlock: HParmBlkPtr): OSErr;
006870   INLINE $205F,$7034,$A660,$3E80;
006871  FUNCTION PBHMapName(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006872  FUNCTION PBHMapNameSync(paramBlock: HParmBlkPtr): OSErr;
006873   INLINE $205F,$7035,$A260,$3E80;
006874  FUNCTION PBHMapNameAsync(paramBlock: HParmBlkPtr): OSErr;
006875   INLINE $205F,$7035,$A660,$3E80;
006876  FUNCTION PBHCopyFile(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006877  FUNCTION PBHCopyFileSync(paramBlock: HParmBlkPtr): OSErr;
006878   INLINE $205F,$7036,$A260,$3E80;
006879  FUNCTION PBHCopyFileAsync(paramBlock: HParmBlkPtr): OSErr;
006880   INLINE $205F,$7036,$A660,$3E80;
006881  FUNCTION PBHMoveRename(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006882  FUNCTION PBHMoveRenameSync(paramBlock: HParmBlkPtr): OSErr;
006883   INLINE $205F,$7037,$A260,$3E80;
006884  FUNCTION PBHMoveRenameAsync(paramBlock: HParmBlkPtr): OSErr;
006885   INLINE $205F,$7037,$A660,$3E80;
006886  FUNCTION PBHOpenDeny(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006887  FUNCTION PBHOpenDenySync(paramBlock: HParmBlkPtr): OSErr;
006888   INLINE $205F,$7038,$A260,$3E80;
006889  FUNCTION PBHOpenDenyAsync(paramBlock: HParmBlkPtr): OSErr;
006890   INLINE $205F,$7038,$A660,$3E80;
006891  FUNCTION PBHOpenRFDeny(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
```

```
006892  FUNCTION PBHOpenRFDenySync(paramBlock: HParmBlkPtr): OSErr;
006893   INLINE $205F,$7039,$A260,$3E80;
006894  FUNCTION PBHOpenRFDenyAsync(paramBlock: HParmBlkPtr): OSErr;
006895   INLINE $205F,$7039,$A660,$3E80;
006896
006897  FUNCTION PBExchangeFiles(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006898  FUNCTION PBExchangeFilesSync(paramBlock: HParmBlkPtr): OSErr;
006899   INLINE $205F,$7017,$A260,$3E80;
006900  FUNCTION PBExchangeFilesAsync(paramBlock: HParmBlkPtr): OSErr;
006901   INLINE $205F,$7017,$A660,$3E80;
006902  FUNCTION PBCreateFileIDRef(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006903  FUNCTION PBCreateFileIDRefSync(paramBlock: HParmBlkPtr): OSErr;
006904   INLINE $205F,$7014,$A260,$3E80;
006905  FUNCTION PBCreateFileIDRefAsync(paramBlock: HParmBlkPtr): OSErr;
006906   INLINE $205F,$7014,$A660,$3E80;
006907  FUNCTION PBResolveFileIDRef(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006908  FUNCTION PBResolveFileIDRefSync(paramBlock: HParmBlkPtr): OSErr;
006909   INLINE $205F,$7016,$A260,$3E80;
006910  FUNCTION PBResolveFileIDRefAsync(paramBlock: HParmBlkPtr): OSErr;
006911   INLINE $205F,$7016,$A660,$3E80;
006912  FUNCTION PBDeleteFileIDRef(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006913  FUNCTION PBDeleteFileIDRefSync(paramBlock: HParmBlkPtr): OSErr;
006914   INLINE $205F,$7015,$A260,$3E80;
006915  FUNCTION PBDeleteFileIDRefAsync(paramBlock: HParmBlkPtr): OSErr;
006916   INLINE $205F,$7015,$A660,$3E80;
006917
006918  FUNCTION PBGetForeignPrivs(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006919  FUNCTION PBGetForeignPrivsSync(paramBlock: HParmBlkPtr): OSErr;
006920   INLINE $205F,$7060,$A260,$3E80;
006921  FUNCTION PBGetForeignPrivsAsync(paramBlock: HParmBlkPtr): OSErr;
006922   INLINE $205F,$7060,$A660,$3E80;
006923  FUNCTION PBSetForeignPrivs(paramBlock: HParmBlkPtr;async: BOOLEAN): OSErr;
006924  FUNCTION PBSetForeignPrivsSync(paramBlock: HParmBlkPtr): OSErr;
006925   INLINE $205F,$7061,$A260,$3E80;
006926  FUNCTION PBSetForeignPrivsAsync(paramBlock: HParmBlkPtr): OSErr;
006927   INLINE $205F,$7061,$A660,$3E80;
006928
006929  {  Desktop Manager  }
006930  FUNCTION PBDTGetPath(paramBlock: DTPBPtr): OSErr;
006931   INLINE $205F,$7020,$A260,$3E80;
006932  FUNCTION PBDTCloseDown(paramBlock: DTPBPtr): OSErr;
006933   INLINE $205F,$7021,$A260,$3E80;
006934  FUNCTION PBDTAddIcon(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006935  FUNCTION PBDTAddIconSync(paramBlock: DTPBPtr): OSErr;
006936   INLINE $205F,$7022,$A260,$3E80;
006937  FUNCTION PBDTAddIconAsync(paramBlock: DTPBPtr): OSErr;
006938   INLINE $205F,$7022,$A660,$3E80;
006939  FUNCTION PBDTGetIcon(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006940  FUNCTION PBDTGetIconSync(paramBlock: DTPBPtr): OSErr;
006941   INLINE $205F,$7023,$A260,$3E80;
006942  FUNCTION PBDTGetIconAsync(paramBlock: DTPBPtr): OSErr;
006943   INLINE $205F,$7023,$A660,$3E80;
006944  FUNCTION PBDTGetIconInfo(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006945  FUNCTION PBDTGetIconInfoSync(paramBlock: DTPBPtr): OSErr;
006946   INLINE $205F,$7024,$A260,$3E80;
006947  FUNCTION PBDTGetIconInfoAsync(paramBlock: DTPBPtr): OSErr;
```

```
006948   INLINE $205F,$7024,$A660,$3E80;
006949  FUNCTION PBDTAddAPPL(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006950  FUNCTION PBDTAddAPPLSync(paramBlock: DTPBPtr): OSErr;
006951   INLINE $205F,$7025,$A260,$3E80;
006952  FUNCTION PBDTAddAPPLAsync(paramBlock: DTPBPtr): OSErr;
006953   INLINE $205F,$7025,$A660,$3E80;
006954  FUNCTION PBDTRemoveAPPL(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006955  FUNCTION PBDTRemoveAPPLSync(paramBlock: DTPBPtr): OSErr;
006956   INLINE $205F,$7026,$A260,$3E80;
006957  FUNCTION PBDTRemoveAPPLAsync(paramBlock: DTPBPtr): OSErr;
006958   INLINE $205F,$7026,$A660,$3E80;
006959  FUNCTION PBDTGetAPPL(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006960  FUNCTION PBDTGetAPPLSync(paramBlock: DTPBPtr): OSErr;
006961   INLINE $205F,$7027,$A260,$3E80;
006962  FUNCTION PBDTGetAPPLAsync(paramBlock: DTPBPtr): OSErr;
006963   INLINE $205F,$7027,$A660,$3E80;
006964  FUNCTION PBDTSetComment(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006965  FUNCTION PBDTSetCommentSync(paramBlock: DTPBPtr): OSErr;
006966   INLINE $205F,$7028,$A260,$3E80;
006967  FUNCTION PBDTSetCommentAsync(paramBlock: DTPBPtr): OSErr;
006968   INLINE $205F,$7028,$A660,$3E80;
006969  FUNCTION PBDTRemoveComment(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006970  FUNCTION PBDTRemoveCommentSync(paramBlock: DTPBPtr): OSErr;
006971   INLINE $205F,$7029,$A260,$3E80;
006972  FUNCTION PBDTRemoveCommentAsync(paramBlock: DTPBPtr): OSErr;
006973   INLINE $205F,$7029,$A660,$3E80;
006974  FUNCTION PBDTGetComment(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006975  FUNCTION PBDTGetCommentSync(paramBlock: DTPBPtr): OSErr;
006976   INLINE $205F,$702A,$A260,$3E80;
006977  FUNCTION PBDTGetCommentAsync(paramBlock: DTPBPtr): OSErr;
006978   INLINE $205F,$702A,$A660,$3E80;
006979  FUNCTION PBDTFlush(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006980  FUNCTION PBDTFlushSync(paramBlock: DTPBPtr): OSErr;
006981   INLINE $205F,$702B,$A260,$3E80;
006982  FUNCTION PBDTFlushAsync(paramBlock: DTPBPtr): OSErr;
006983   INLINE $205F,$702B,$A660,$3E80;
006984  FUNCTION PBDTReset(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006985  FUNCTION PBDTResetSync(paramBlock: DTPBPtr): OSErr;
006986   INLINE $205F,$702C,$A260,$3E80;
006987  FUNCTION PBDTResetAsync(paramBlock: DTPBPtr): OSErr;
006988   INLINE $205F,$702C,$A660,$3E80;
006989  FUNCTION PBDTGetInfo(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006990  FUNCTION PBDTGetInfoSync(paramBlock: DTPBPtr): OSErr;
006991   INLINE $205F,$702D,$A260,$3E80;
006992  FUNCTION PBDTGetInfoAsync(paramBlock: DTPBPtr): OSErr;
006993   INLINE $205F,$702D,$A660,$3E80;
006994  FUNCTION PBDTOpenInform(paramBlock: DTPBPtr): OSErr;
006995   INLINE $205F,$702E,$A060,$3E80;
006996  FUNCTION PBDTDelete(paramBlock: DTPBPtr;async: BOOLEAN): OSErr;
006997  FUNCTION PBDTDeleteSync(paramBlock: DTPBPtr): OSErr;
006998   INLINE $205F,$702F,$A060,$3E80;
006999  FUNCTION PBDTDeleteAsync(paramBlock: DTPBPtr): OSErr;
007000   INLINE $205F,$702F,$A460,$3E80;
007001
007002  {  VolumeMount traps  }
007003  FUNCTION PBGetVolMountInfoSize(paramBlock: ParmBlkPtr): OSErr;
```

```
007004   INLINE $205F,$703F,$A260,$3E80;
007005  FUNCTION PBGetVolMountInfo(paramBlock: ParmBlkPtr): OSErr;
007006   INLINE $205F,$7040,$A260,$3E80;
007007  FUNCTION PBVolumeMount(paramBlock: ParmBlkPtr): OSErr;
007008   INLINE $205F,$7041,$A260,$3E80;
007009
007010  {  FSp traps  }
007011  FUNCTION FSMakeFSSpec(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;
007012   VAR spec: FSSpec): OSErr;
007013   INLINE $303C, $0001, $AA52;
007014  FUNCTION FSpOpenDF(spec: FSSpec;permission: SignedByte;VAR refNum: INTEGER): OSErr;
007015   INLINE $303C, $0002, $AA52;
007016  FUNCTION FSpOpenRF(spec: FSSpec ;permission: SignedByte;VAR refNum: INTEGER): OSErr;
007017   INLINE $303C, $0003, $AA52;
007018  FUNCTION FSpCreate(spec: FSSpec ;creator: OSType;fileType: OSType;scriptTag: ScriptCode): OSErr;
007019   INLINE $303C, $0004, $AA52;
007020  FUNCTION FSpDirCreate(spec: FSSpec;scriptTag: ScriptCode;VAR createdDirID: LONGINT): OSErr;
007021   INLINE $303C, $0005, $AA52;
007022  FUNCTION FSpDelete(spec: FSSpec): OSErr;
007023   INLINE $303C, $0006, $AA52;
007024  FUNCTION FSpGetFInfo(spec: FSSpec;VAR fndrInfo: FInfo): OSErr;
007025   INLINE $303C, $0007, $AA52;
007026  FUNCTION FSpSetFInfo(spec: FSSpec;fndrInfo: FInfo): OSErr;
007027   INLINE $303C, $0008, $AA52;
007028  FUNCTION FSpSetFLock(spec: FSSpec): OSErr;
007029   INLINE $303C, $0009, $AA52;
007030  FUNCTION FSpRstFLock(spec: FSSpec): OSErr;
007031   INLINE $303C, $000A, $AA52;
007032  FUNCTION FSpRename(spec: FSSpec;newName: Str255): OSErr;
007033   INLINE $303C, $000B, $AA52;
007034  FUNCTION FSpCatMove(source: FSSpec;dest: FSSpec): OSErr;
007035   INLINE $303C, $000C, $AA52;
007036  FUNCTION FSpExchangeFiles(source: FSSpec;dest: FSSpec): OSErr;
007037   INLINE $303C, $000F, $AA52;
007038
007039
007040  {$ENDC} { UsingFiles }
007041
007042  {$IFC NOT UsingIncludes}
007043   END.
007044  {$ENDC}
007045
007046
007047  ### END OF FILE Files.p
007048
```

```
007049
007050  ##########################################################################
007051  ### FILE: FileTransfers.p
007052  ##########################################################################
007053
007054
007055  {
007056  Created: Wednesday, September 11, 1991 at 6:08 PM
007057   FileTransfers.p
007058   Pascal Interface to the Macintosh Libraries
007059
007060    Copyright Apple Computer, Inc. 1988-1991
007061    All rights reserved
007062  }
007063
007064
007065  {$IFC UNDEFINED UsingIncludes}
007066  {$SETC UsingIncludes := 0}
007067  {$ENDC}
007068
007069  {$IFC NOT UsingIncludes}
007070   UNIT FileTransfers;
007071   INTERFACE
007072  {$ENDC}
007073
007074  {$IFC UNDEFINED UsingFileTransfers}
007075  {$SETC UsingFileTransfers := 1}
007076
007077  {$I+}
007078  {$SETC FileTransfersIncludes := UsingIncludes}
007079  {$SETC UsingIncludes := 1}
007080  {$IFC UNDEFINED UsingPackages}
007081  {$I $$Shell(PInterfaces)Packages.p}
007082  {$ENDC}
007083  {$IFC UNDEFINED UsingCTBUtilities}
007084  {$I $$Shell(PInterfaces)CTBUtilities.p}
007085  {$ENDC}
007086  {$IFC UNDEFINED UsingConnections}
007087  {$I $$Shell(PInterfaces)Connections.p}
007088  {$ENDC}
007089  {$IFC UNDEFINED UsingFiles}
007090  {$I $$Shell(PInterfaces)Files.p}
007091  {$ENDC}
007092  {$SETC UsingIncludes := FileTransfersIncludes}
007093
007094  CONST
007095
007096  { current file transfer manager version }
007097  curFTVersion = 2;
007098
007099  { FTErr    }
007100  ftGenericError = -1;
007101  ftNoErr = 0;
007102  ftRejected = 1;
007103  ftFailed = 2;
007104  ftTimeOut = 3;
```

```
007105  ftTooManyRetry = 4;
007106  ftNotEnoughDSpace = 5;
007107  ftRemoteCancel = 6;
007108  ftWrongFormat = 7;
007109  ftNoTools = 8;
007110  ftUserCancel = 9;
007111  ftNotSupported = 10;
007112
007113  TYPE
007114  FTErr = OSErr;
007115
007116  CONST
007117
007118  { FTFlags }
007119  ftIsFTMode = $00000001;
007120  ftNoMenus = $00000002;
007121  ftQuiet = $00000004;
007122  ftConfigChanged = $00000010;
007123  ftSucc = $00000080;
007124
007125  TYPE
007126  FTFlags = LONGINT;
007127
007128
007129  CONST
007130
007131  { FTAttributes }
007132  ftSameCircuit = $0001;
007133  ftSendDisable = $0002;
007134  ftReceiveDisable = $0004;
007135  ftTextOnly = $0008;
007136  ftNoStdFile = $0010;
007137  ftMultipleFileSend = $0020;
007138
007139  TYPE
007140  FTAttributes = INTEGER;
007141
007142
007143  CONST
007144
007145  { FTDirection }
007146  ftReceiving = 0;
007147  ftTransmitting = 1;
007148
007149  TYPE
007150  FTDirection = INTEGER;
007151
007152
007153  FTPtr = ^FTRecord;
007154  FTHandle = ^FTPtr;
007155  FTRecord = PACKED RECORD
007156   procID: INTEGER;
007157   flags: FTFlags;
007158   errCode: FTErr;
007159   refCon: LONGINT;
007160   userData: LONGINT;
```

```
007161   defProc: ProcPtr;
007162   config: Ptr;
007163   oldConfig: Ptr;
007164   environsProc: ProcPtr;
007165   reserved1: LONGINT;
007166   reserved2: LONGINT;
007167   ftPrivate: Ptr;
007168   sendProc: ProcPtr;
007169   recvProc: ProcPtr;
007170   writeProc: ProcPtr;
007171   readProc: ProcPtr;
007172   owner: WindowPtr;
007173   direction: FTDirection;
007174   theReply: SFReply;
007175   writePtr: LONGINT;
007176   readPtr: LONGINT;
007177   theBuf: ^char;
007178   bufSize: LONGINT;
007179   autoRec: Str255;
007180   attributes: FTAttributes;
007181   END;
007182
007183
007184  CONST
007185
007186  { FTReadProc messages }
007187  ftReadOpenFile = 0;     { count = forkFlags, buffer = pblock from PBGetFInfo }
007188  ftReadDataFork = 1;
007189  ftReadRsrcFork = 2;
007190  ftReadAbort = 3;
007191  ftReadComplete = 4;
007192  ftReadSetFPos = 6;      { count = forkFlags, buffer = pBlock same as PBSetFPos }
007193  ftReadGetFPos = 7;      { count = forkFlags, buffer = pBlock same as PBGetFPos }
007194
007195  { FTWriteProc messages }
007196  ftWriteOpenFile = 0;    { count = forkFlags, buffer = pblock from PBGetFInfo }
007197  ftWriteDataFork = 1;
007198  ftWriteRsrcFork = 2;
007199  ftWriteAbort = 3;
007200  ftWriteComplete = 4;
007201  ftWriteFileInfo = 5;
007202  ftWriteSetFPos = 6;     { count = forkFlags, buffer = pBlock same as PBSetFPos }
007203  ftWriteGetFPos = 7;     { count = forkFlags, buffer = pBlock same as PBGetFPos }
007204
007205  { fork flags }
007206  ftOpenDataFork = 1;
007207  ftOpenRsrcFork = 2;
007208
007209
007210  TYPE
007211  { application routines type definitions }
007212  FileTransferReadProcPtr = ProcPtr;
007213  FileTransferWriteProcPtr = ProcPtr;
007214
007215  FileTransferSendProcPtr = ProcPtr;
007216  FileTransferReceiveProcPtr = ProcPtr;
```

```
007217
007218  FileTransferEnvironsProcPtr = ProcPtr;
007219
007220  FileTransferNotificationProcPtr = ProcPtr;
007221  FileTransferChooseIdleProcPtr = ProcPtr;
007222
007223  FUNCTION InitFT: FTErr;
007224  FUNCTION FTGetVersion(hFT: FTHandle): Handle;
007225  FUNCTION FTGetFTVersion: INTEGER;
007226
007227  FUNCTION FTNew(procID: INTEGER;flags: FTFlags;sendProc: FileTransferSendProcPtr;
007228   recvProc: FileTransferReceiveProcPtr;readProc: FileTransferReadProcPtr;
007229   writeProc: FileTransferWriteProcPtr;environsProc: FileTransferEnvironsProcPtr;
007230   owner: WindowPtr;refCon: LONGINT;userData: LONGINT): FTHandle;
007231
007232  PROCEDURE FTDispose(hFT: FTHandle);
007233
007234  FUNCTION FTStart(hFT: FTHandle;direction: FTDirection;fileInfo: SFReply): FTErr;
007235  FUNCTION FTAbort(hFT: FTHandle): FTErr;
007236
007237  FUNCTION FTSend(hFT: FTHandle;numFiles: INTEGER;pFSSpec: FSSpecArrayPtr;
007238   notifyProc: FileTransferNotificationProcPtr): FTErr;
007239  FUNCTION FTReceive(hFT: FTHandle;pFSSpec: FSSpecPtr;notifyProc: FileTransferNotificationProcPtr): FTErr;
007240
007241  PROCEDURE FTExec(hFT: FTHandle);
007242
007243  PROCEDURE FTActivate(hFT: FTHandle;activate: BOOLEAN);
007244  PROCEDURE FTResume(hFT: FTHandle;resume: BOOLEAN);
007245  FUNCTION FTMenu(hFT: FTHandle;menuID: INTEGER;item: INTEGER): BOOLEAN;
007246
007247  FUNCTION FTChoose(VAR hFT: FTHandle;where: Point;idleProc: FileTransferChooseIdleProcPtr): INTEGER;
007248  PROCEDURE FTEvent(hFT: FTHandle;theEvent: EventRecord);
007249
007250  FUNCTION FTValidate(hFT: FTHandle): BOOLEAN;
007251  PROCEDURE FTDefault(VAR theConfig: Ptr;procID: INTEGER;allocate: BOOLEAN);
007252
007253  FUNCTION FTSetupPreflight(procID: INTEGER;VAR magicCookie: LONGINT): Handle;
007254  PROCEDURE FTSetupSetup(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
007255   VAR magicCookie: LONGINT);
007256  FUNCTION FTSetupFilter(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
007257   VAR theEvent: EventRecord;VAR theItem: INTEGER;VAR magicCookie: LONGINT): BOOLEAN;
007258  PROCEDURE FTSetupItem(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
007259   VAR theItem: INTEGER;VAR magicCookie: LONGINT);
007260  PROCEDURE FTSetupXCleanup(procID: INTEGER;theConfig: Ptr;count: INTEGER;
007261   theDialog: DialogPtr;OKed: BOOLEAN;VAR magicCookie: LONGINT);
007262
007263  PROCEDURE FTSetupPostflight(procID: INTEGER);
007264
007265  FUNCTION FTGetConfig(hFT: FTHandle): Ptr;
007266  FUNCTION FTSetConfig(hFT: FTHandle;thePtr: Ptr): INTEGER;
007267
007268  FUNCTION FTIntlToEnglish(hFT: FTHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
007269   language: INTEGER): OSErr;
007270  FUNCTION FTEnglishToIntl(hFT: FTHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
007271   language: INTEGER): OSErr;
007272
```

```
007273  PROCEDURE FTGetToolName(procID: INTEGER;VAR name: Str255);
007274  FUNCTION FTGetProcID(name: Str255): INTEGER;
007275
007276  PROCEDURE FTSetRefCon(hFT: FTHandle;refCon: LONGINT);
007277  FUNCTION FTGetRefCon(hFT: FTHandle): LONGINT;
007278
007279  PROCEDURE FTSetUserData(hFT: FTHandle;userData: LONGINT);
007280  FUNCTION FTGetUserData(hFT: FTHandle): LONGINT;
007281
007282  PROCEDURE FTGetErrorString(hFT: FTHandle;id: INTEGER;VAR errMsg: Str255);
007283
007284
007285  {$ENDC} { UsingFileTransfers }
007286
007287  {$IFC NOT UsingIncludes}
007288   END.
007289  {$ENDC}
007290
007291
007292  ### END OF FILE FileTransfers.p
007293
```

```
007294
007295  ############################################################################
007296  ### FILE: FileTransferTools.p
007297  ############################################################################
007298
007299
007300  {
007301  Created: Thursday, September 12, 1991 at 9:51 AM
007302   FileTransferTools.p
007303   Pascal Interface to the Macintosh Libraries
007304
007305    Copyright Apple Computer, Inc. 1988-1991
007306    All rights reserved
007307  }
007308
007309
007310  {$IFC UNDEFINED UsingIncludes}
007311  {$SETC UsingIncludes := 0}
007312  {$ENDC}
007313
007314  {$IFC NOT UsingIncludes}
007315   UNIT FileTransferTools;
007316   INTERFACE
007317  {$ENDC}
007318
007319  {$IFC UNDEFINED UsingFileTransferTools}
007320  {$SETC UsingFileTransferTools := 1}
007321
007322  {$I+}
007323  {$SETC FileTransferToolsIncludes := UsingIncludes}
007324  {$SETC UsingIncludes := 1}
007325  {$IFC UNDEFINED UsingDialogs}
007326  {$I $$Shell(PInterfaces)Dialogs.p}
007327  {$ENDC}
007328  {$IFC UNDEFINED UsingFileTransfers}
007329  {$I $$Shell(PInterfaces)FileTransfers.p}
007330  {$ENDC}
007331  {$SETC UsingIncludes := FileTransferToolsIncludes}
007332
007333  CONST
007334
007335  { Control }
007336  ftInitMsg = 0;
007337  ftDisposeMsg = 1;
007338  ftSuspendMsg = 2;
007339  ftResumeMsg = 3;
007340  ftMenuMsg = 4;
007341  ftEventMsg = 5;
007342  ftActivateMsg = 6;
007343  ftDeactivateMsg = 7;
007344  ftGetErrorStringMsg = 8;
007345
007346  ftAbortMsg = 52;
007347
007348  ftStartMsg = 100;
007349  ftExecMsg = 102;
```

```
007350  ftSendMsg = 103;
007351  ftReceiveMsg = 104;
007352
007353  {Setup }
007354  ftSpreflightMsg = 0;
007355  ftSsetupMsg = 1;
007356  ftSitemMsg = 2;
007357  ftSfilterMsg = 3;
007358  ftScleanupMsg = 4;
007359
007360  { validate }
007361  ftValidateMsg = 0;
007362  ftDefaultMsg = 1;
007363
007364  { scripting }
007365  ftMgetMsg = 0;
007366  ftMsetMsg = 1;
007367
007368  { localization }
007369  ftL2English = 0;
007370  ftL2Intl = 1;
007371
007372  { DEFs }
007373  fdefType = 'fdef';
007374  fsetType = 'fset';
007375  fvalType = 'fval';
007376  flocType = 'floc';
007377  fscrType = 'fscr';
007378
007379  fbndType = 'fbnd';
007380  fverType = 'vers';
007381
007382  TYPE
007383  FTSetupPtr = ^FTSetupStruct;
007384  FTSetupStruct = PACKED RECORD
007385   theDialog: DialogPtr;  { the dialog form the application }
007386   count: INTEGER;        { first appended item }
007387   theConfig: Ptr;        { the config record to setup }
007388   procID: INTEGER;       { procID of the tool }
007389   END;
007390
007391
007392
007393  {$ENDC} { UsingFileTransferTools }
007394
007395  {$IFC NOT UsingIncludes}
007396   END.
007397  {$ENDC}
007398
007399
007400  ### END OF FILE FileTransferTools.p
007401
```

```
007402
007403  ##########################################################################
007404  ### FILE: Finder.p
007405  ##########################################################################
007406
007407
007408  {
007409  Created: Tuesday, November 26, 1991 at 11:43 AM
007410   Finder.p
007411   Pascal Interface to the Macintosh Libraries
007412
007413    Copyright Apple Computer, Inc. 1990-1991
007414    All rights reserved
007415  }
007416
007417
007418  {$IFC UNDEFINED UsingIncludes}
007419  {$SETC UsingIncludes := 0}
007420  {$ENDC}
007421
007422  {$IFC NOT UsingIncludes}
007423   UNIT Finder;
007424   INTERFACE
007425  {$ENDC}
007426
007427  {$IFC UNDEFINED UsingFinder}
007428  {$SETC UsingFinder := 1}
007429
007430
007431  CONST
007432
007433  { make only the following consts avaiable to resource files that include this file }
007434  kCustomIconResource = -16455;           { Custom icon family resource ID }
007435
007436  kContainerFolderAliasType = 'fdrp';     { type for folder aliases }
007437  kContainerTrashAliasType = 'trsh';      { type for trash folder aliases }
007438  kContainerHardDiskAliasType = 'hdsk';   { type for hard disk aliases }
007439  kContainerFloppyAliasType = 'flpy';     { type for floppy aliases }
007440  kContainerServerAliasType = 'srvr';     { type for server aliases }
007441  kApplicationAliasType = 'adrp';         { type for application aliases }
007442  kContainerAliasType = 'drop';           { type for all other containers }
007443
007444  { type for Special folder aliases }
007445  kSystemFolderAliasType = 'fasy';
007446  kAppleMenuFolderAliasType = 'faam';
007447  kStartupFolderAliasType = 'fast';
007448  kPrintMonitorDocsFolderAliasType = 'fapn';
007449  kPreferencesFolderAliasType = 'fapf';
007450  kControlPanelFolderAliasType = 'fact';
007451  kExtensionFolderAliasType = 'faex';
007452
007453  { type for AppleShare folder aliases }
007454  kExportedFolderAliasType = 'faet';
007455  kDropFolderAliasType = 'fadr';
007456  kSharedFolderAliasType = 'fash';
007457  kMountedFolderAliasType = 'famn';
```

```
007458
007459
007460  {Finder Flags}
007461  kIsOnDesk = $1;
007462  kColor = $E;
007463
007464  {kColorReserved = $10
007465  kRequiresSwitchLaunch = $20}
007466
007467  kIsShared = $40;
007468
007469  {kHasNoINITs = $80}
007470
007471  kHasBeenInited = $100;
007472
007473  {kReserved = $200}
007474
007475  kHasCustomIcon = $400;
007476  kIsStationary = $800;
007477  kNameLocked = $1000;
007478  kHasBundle = $2000;
007479  kIsInvisible = $4000;
007480  kIsAlias = $8000;
007481
007482
007483
007484  {$ENDC} { UsingFinder }
007485
007486  {$IFC NOT UsingIncludes}
007487   END.
007488  {$ENDC}
007489
007490
007491  ### END OF FILE Finder.p
007492
```

```
007493
007494  ############################################################################
007495  ### FILE: FixMath.p
007496  ############################################################################
007497
007498  {
007499  Created: Thursday, March 15, 1990 at 8:43 AM
007500      FixMath.p
007501      Pascal Interface to Fixed Point Math
007502
007503      Copyright Apple Computer, Inc.  1985-1990
007504      All rights reserved
007505
007506  CHANGE LOG:
007507      23 Oct 90  JPO  Changed functions Frac2X, Fix2X, X2Fix, and X2Frac
007508                        to in-line trap calls for non-mc68881 mode
007509  }
007510
007511
007512  {$IFC UNDEFINED UsingIncludes}
007513  {$SETC UsingIncludes := 0}
007514  {$ENDC}
007515
007516  {$IFC NOT UsingIncludes}
007517      UNIT FixMath;
007518      INTERFACE
007519  {$ENDC}
007520
007521  {$IFC UNDEFINED UsingFixMath}
007522  {$SETC UsingFixMath := 1}
007523
007524  {$I+}
007525  {$SETC FixMathIncludes := UsingIncludes}
007526  {$SETC UsingIncludes := 1}
007527  {$IFC UNDEFINED UsingTypes}
007528  {$I $$Shell(PInterfaces)Types.p}
007529  {$ENDC}
007530  {$SETC UsingIncludes := FixMathIncludes}
007531
007532
007533  FUNCTION Fix2Frac(x: Fixed): Fract;
007534      INLINE $A841;
007535  FUNCTION Fix2Long(x: Fixed): LONGINT;
007536      INLINE $A840;
007537  FUNCTION FixATan2(x: LONGINT;y: LONGINT): Fixed;
007538      INLINE $A818;
007539  FUNCTION Long2Fix(x: LONGINT): Fixed;
007540      INLINE $A83F;
007541  FUNCTION Frac2Fix(x: Fract): Fixed;
007542      INLINE $A842;
007543
007544  {$IFC OPTION(MC68881)}
007545
007546  FUNCTION Frac2X(x: Fract): Extended;
007547  FUNCTION Fix2X(x: Fixed): Extended;
007548  FUNCTION X2Fix(x: Extended): Fixed;
```

```
007549  FUNCTION X2Frac(x: Extended): Fract;
007550
007551  {$ELSEC}
007552
007553  FUNCTION Frac2X(x: Fract): Extended;
007554      INLINE $A845;
007555  FUNCTION Fix2X(x: Fixed): Extended;
007556      INLINE $A843;
007557  FUNCTION X2Fix(x: Extended): Fixed;
007558      INLINE $A844;
007559  FUNCTION X2Frac(x: Extended): Fract;
007560      INLINE $A846;
007561
007562  {$ENDC}
007563
007564  FUNCTION FracMul(x: Fract;y: Fract): Fract;
007565      INLINE $A84A;
007566  FUNCTION FixDiv(x: Fixed;y: Fixed): Fixed;
007567      INLINE $A84D;
007568  FUNCTION FracDiv(x: Fract;y: Fract): Fract;
007569      INLINE $A84B;
007570  FUNCTION FracSqrt(x: Fract): Fract;
007571      INLINE $A849;
007572  FUNCTION FracSin(x: Fixed): Fract;
007573      INLINE $A848;
007574  FUNCTION FracCos(x: Fixed): Fract;
007575      INLINE $A847;
007576
007577  {$ENDC}    { UsingFixMath }
007578
007579  {$IFC NOT UsingIncludes}
007580      END.
007581  {$ENDC}
007582
007583
007584  ### END OF FILE FixMath.p
007585
```

```
007586
007587  ############################################################################
007588  ### FILE: Folders.p
007589  ############################################################################
007590
007591  {
007592  Created: Sunday, January 6, 1991 at 10:43 PM
007593      Folders.p
007594      Pascal Interface to the Macintosh Libraries
007595
007596          Copyright Apple Computer, Inc.    1989-90
007597          All rights reserved
007598  }
007599
007600
007601  {$IFC UNDEFINED UsingIncludes}
007602  {$SETC UsingIncludes := 0}
007603  {$ENDC}
007604
007605  {$IFC NOT UsingIncludes}
007606      UNIT Folders;
007607      INTERFACE
007608  {$ENDC}
007609
007610  {$IFC UNDEFINED UsingFolders}
007611  {$SETC UsingFolders := 1}
007612
007613  {$I+}
007614  {$SETC FoldersIncludes := UsingIncludes}
007615  {$SETC UsingIncludes := 1}
007616  {$IFC UNDEFINED UsingTypes}
007617  {$I $$Shell(PInterfaces)Types.p}
007618  {$ENDC}
007619  {$IFC UNDEFINED UsingFiles}
007620  {$I $$Shell(PInterfaces)Files.p}
007621  {$ENDC}
007622  {$SETC UsingIncludes := FoldersIncludes}
007623
007624  CONST
007625  kOnSystemDisk = $8000;
007626
007627  kCreateFolder = TRUE;
007628  kDontCreateFolder = FALSE;
007629
007630  kSystemFolderType = 'macs';             {the system folder}
007631  kDesktopFolderType = 'desk';            {the desktop folder; objects in this folder show on the desk top.}
007632  kTrashFolderType = 'trsh';              {the trash folder; objects in this folder show up in the trash}
007633  kWhereToEmptyTrashFolderType = 'empt';  {the "empty trash" folder; Finder starts empty from here down}
007634
007635  kPrintMonitorDocsFolderType = 'prnt';   { Print Monitor documents }
007636
007637  kStartupFolderType = 'strt';            {Finder objects (applications, documents, DAs, aliases, to...) to open at startup go here}
007638  kAppleMenuFolderType = 'amnu';          {Finder objects to put into the Apple menu go here}
007639  kControlPanelFolderType = 'ctrl';       {Control Panels go here (may contain INITs)}
007640  kExtensionFolderType = 'extn';          {Finder extensions go here}
007641
```

```
007642  kPreferencesFolderType = 'pref';        {preferences for applications go here}
007643  kTemporaryFolderType = 'temp';          {temporary files go here (deleted periodically, but don't rely on it.)}
007644
007645  FUNCTION FindFolder(vRefNum: INTEGER;folderType: OSType;createFolder: BOOLEAN;
007646      VAR foundVRefNum: INTEGER;VAR foundDirID: LONGINT): OSErr;
007647  {$IFC SystemSevenOrLater }
007648      INLINE $7000,$A823;
007649  {$ENDC}
007650
007651
007652  {$ENDC}    { UsingFolders }
007653
007654  {$IFC NOT UsingIncludes}
007655      END.
007656  {$ENDC}
007657
007658
007659  ### END OF FILE Folders.p
007660
```

```
007661
007662  ##########################################################################
007663  ### FILE: Fonts.p
007664  ##########################################################################
007665
007666  {
007667  Created: Monday, January 28, 1991 at 4:48 PM
007668      Fonts.p
007669      Pascal Interface to the Macintosh Libraries
007670
007671          Copyright Apple Computer, Inc.    1985-1990
007672          All rights reserved
007673  }
007674
007675
007676  {$IFC UNDEFINED UsingIncludes}
007677  {$SETC UsingIncludes := 0}
007678  {$ENDC}
007679
007680  {$IFC NOT UsingIncludes}
007681      UNIT Fonts;
007682      INTERFACE
007683  {$ENDC}
007684
007685  {$IFC UNDEFINED UsingFonts}
007686  {$SETC UsingFonts := 1}
007687
007688  {$I+}
007689  {$SETC FontsIncludes := UsingIncludes}
007690  {$SETC UsingIncludes := 1}
007691  {$IFC UNDEFINED UsingTypes}
007692  {$I $$Shell(PInterfaces)Types.p}
007693  {$ENDC}
007694  {$IFC UNDEFINED UsingQuickdraw}
007695  {$I $$Shell(PInterfaces)Quickdraw.p}
007696  {$ENDC}
007697  {$SETC UsingIncludes := FontsIncludes}
007698
007699  CONST
007700  systemFont = 0;
007701  applFont = 1;
007702  newYork = 2;
007703  geneva = 3;
007704  monaco = 4;
007705  venice = 5;
007706  london = 6;
007707  athens = 7;
007708  sanFran = 8;
007709  toronto = 9;
007710  cairo = 11;
007711  losAngeles = 12;
007712  times = 20;
007713  helvetica = 21;
007714  courier = 22;
007715  symbol = 23;
007716  mobile = 24;
```

```
007717  commandMark = 17;
007718  checkMark = 18;
007719  diamondMark = 19;
007720  appleMark = 20;
007721  propFont = 36864;
007722  prpFntH = 36865;
007723  prpFntW = 36866;
007724  prpFntHW = 36867;
007725  fixedFont = 45056;
007726  fxdFntH = 45057;
007727  fxdFntW = 45058;
007728  fxdFntHW = 45059;
007729  fontWid = 44208;
007730
007731  TYPE
007732  FMInput = PACKED RECORD
007733      family: INTEGER;
007734      size: INTEGER;
007735      face: Style;
007736      needBits: BOOLEAN;
007737      device: INTEGER;
007738      numer: Point;
007739      denom: Point;
007740      END;
007741
007742  FMOutPtr = ^FMOutput;
007743  FMOutput = PACKED RECORD
007744      errNum: INTEGER;
007745      fontHandle: Handle;
007746      bold: Byte;
007747      italic: Byte;
007748      ulOffset: Byte;
007749      ulShadow: Byte;
007750      ulThick: Byte;
007751      shadow: Byte;
007752      extra: SignedByte;
007753      ascent: Byte;
007754      descent: Byte;
007755      widMax: Byte;
007756      leading: SignedByte;
007757      unused: Byte;
007758      numer: Point;
007759      denom: Point;
007760      END;
007761
007762  FontRec = RECORD
007763      fontType: INTEGER;                  {font type}
007764      firstChar: INTEGER;                 {ASCII code of first character}
007765      lastChar: INTEGER;                  {ASCII code of last character}
007766      widMax: INTEGER;                    {maximum character width}
007767      kernMax: INTEGER;                   {negative of maximum character kern}
007768      nDescent: INTEGER;                  {negative of descent}
007769      fRectWidth: INTEGER;                {width of font rectangle}
007770      fRectHeight: INTEGER;               {height of font rectangle}
007771      owTLoc: INTEGER;                    {offset to offset/width table}
007772      ascent: INTEGER;                    {ascent}
```

```
007773      descent: INTEGER;                       {descent}
007774      leading: INTEGER;                       {leading}
007775      rowWords: INTEGER;                      {row width of bit image / 2 }
007776      END;
007777
007778  FMetricRec = RECORD
007779      ascent: Fixed;                          {base line to top}
007780      descent: Fixed;                         {base line to bottom}
007781      leading: Fixed;                         {leading between lines}
007782      widMax: Fixed;                          {maximum character width}
007783      wTabHandle: Handle;                     {handle to font width table}
007784      END;
007785
007786  WidEntry = RECORD
007787      widStyle: INTEGER;                      {style entry applies to}
007788      END;
007789
007790  WidTable = RECORD
007791      numWidths: INTEGER;                     {number of entries - 1}
007792      END;
007793
007794  AsscEntry = RECORD
007795      fontSize: INTEGER;
007796      fontStyle: INTEGER;
007797      fontID: INTEGER;                        {font resource ID}
007798      END;
007799
007800  FontAssoc = RECORD
007801      numAssoc: INTEGER;                      {number of entries - 1}
007802      END;
007803
007804  StyleTable = RECORD
007805      fontClass: INTEGER;
007806      offset: LONGINT;
007807      reserved: LONGINT;
007808      indexes: ARRAY [0..47] OF SignedByte;
007809      END;
007810
007811  NameTable = RECORD
007812      stringCount: INTEGER;
007813      baseFontName: Str255;
007814      END;
007815
007816  KernPair = RECORD
007817      kernFirst: CHAR;                        {1st character of kerned pair}
007818      kernSecond: CHAR;                       {2nd character of kerned pair}
007819      kernWidth: INTEGER;                     {kerning in 1pt fixed format}
007820      END;
007821
007822  KernEntry = RECORD
007823      kernLength: INTEGER;                    {length of this entry}
007824      kernStyle: INTEGER;                     {style the entry applies to}
007825      END;
007826
007827  KernTable = RECORD
007828      numKerns: INTEGER;                      {number of kerning entries}
```

```
007829      END;
007830
007831  WidthTable = PACKED RECORD
007832      tabData: ARRAY [1..256] OF Fixed;       {character widths}
007833      tabFont: Handle;                        {font record used to build table}
007834      sExtra: LONGINT;                        {space extra used for table}
007835      style: LONGINT;                         {extra due to style}
007836      fID: INTEGER;                           {font family ID}
007837      fSize: INTEGER;                         {font size request}
007838      face: INTEGER;                          {style (face) request}
007839      device: INTEGER;                        {device requested}
007840      inNumer: Point;                         {scale factors requested}
007841      inDenom: Point;                         {scale factors requested}
007842      aFID: INTEGER;                          {actual font family ID for table}
007843      fHand: Handle;                          {family record used to build up table}
007844      usedFam: BOOLEAN;                       {used fixed point family widths}
007845      aFace: Byte;                            {actual face produced}
007846      vOutput: INTEGER;                       {vertical scale output value}
007847      hOutput: INTEGER;                       {horizontal scale output value}
007848      vFactor: INTEGER;                       {vertical scale output value}
007849      hFactor: INTEGER;                       {horizontal scale output value}
007850      aSize: INTEGER;                         {actual size of actual font used}
007851      tabSize: INTEGER;                       {total size of table}
007852      END;
007853
007854  FamRec = RECORD
007855      ffFlags: INTEGER;                       {flags for family}
007856      ffFamID: INTEGER;                       {family ID number}
007857      ffFirstChar: INTEGER;                   {ASCII code of 1st character}
007858      ffLastChar: INTEGER;                    {ASCII code of last character}
007859      ffAscent: INTEGER;                      {maximum ascent for 1pt font}
007860      ffDescent: INTEGER;                     {maximum descent for 1pt font}
007861      ffLeading: INTEGER;                     {maximum leading for 1pt font}
007862      ffWidMax: INTEGER;                      {maximum widMax for 1pt font}
007863      ffWTabOff: LONGINT;                     {offset to width table}
007864      ffKernOff: LONGINT;                     {offset to kerning table}
007865      ffStylOff: LONGINT;                     {offset to style mapping table}
007866      ffProperty: ARRAY [1..9] OF INTEGER;    {style property info}
007867      ffIntl: ARRAY [1..2] OF INTEGER;        {for international use}
007868      ffVersion: INTEGER;                     {version number}
007869      END;
007870
007871
007872  PROCEDURE InitFonts;
007873      INLINE $A8FE;
007874  PROCEDURE GetFontName(familyID: INTEGER;VAR name: Str255);
007875      INLINE $A8FF;
007876  PROCEDURE GetFNum(name: Str255;VAR familyID: INTEGER);
007877      INLINE $A900;
007878  FUNCTION RealFont(fontNum: INTEGER;size: INTEGER): BOOLEAN;
007879      INLINE $A902;
007880  PROCEDURE SetFontLock(lockFlag: BOOLEAN);
007881      INLINE $A903;
007882  FUNCTION FMSwapFont(inRec: FMInput): FMOutPtr;
007883      INLINE $A901;
007884  PROCEDURE SetFScaleDisable(fscaleDisable: BOOLEAN);
```

```
007885      INLINE $A834;
007886  PROCEDURE FontMetrics(theMetrics: FMetricRec);
007887      INLINE $A835;
007888  PROCEDURE SetFractEnable(fractEnable: BOOLEAN);
007889  FUNCTION IsOutline(numer: Point;denom: Point): BOOLEAN;
007890      INLINE $7000,$A854;
007891  PROCEDURE SetOutlinePreferred(outlinePreferred: BOOLEAN);
007892      INLINE $7001,$A854;
007893  FUNCTION GetOutlinePreferred: BOOLEAN;
007894      INLINE $7009,$A854;
007895  FUNCTION OutlineMetrics(byteCount: INTEGER;textPtr: UNIV Ptr;numer: Point;
007896      denom: Point;VAR yMax: INTEGER;VAR yMin: INTEGER;awArray: FixedPtr;lsbArray: FixedPtr;
007897      boundsArray: RectPtr): OSErr;
007898      INLINE $7008,$A854;
007899  PROCEDURE SetPreserveGlyph(preserveGlyph: BOOLEAN);
007900      INLINE $700A,$A854;
007901  FUNCTION GetPreserveGlyph: BOOLEAN;
007902      INLINE $700B,$A854;
007903  FUNCTION FlushFonts: OSErr;
007904      INLINE $700C,$A854;
007905
007906
007907  {$ENDC}    { UsingFonts }
007908
007909  {$IFC NOT UsingIncludes}
007910      END.
007911  {$ENDC}
007912
007913
007914  ### END OF FILE Fonts.p
007915
```

```
007916
007917  ############################################################################
007918  ### FILE: GestaltEqu.p
007919  ############################################################################
007920
007921
007922  {
007923  Created: Wednesday, December 4, 1991 at 12:31 PM
007924   GestaltEqu.p
007925   Pascal Interface to the Macintosh Libraries
007926
007927    Copyright Apple Computer, Inc. 1988-1991
007928    All rights reserved
007929  }
007930
007931
007932  {$IFC UNDEFINED UsingIncludes}
007933  {$SETC UsingIncludes := 0}
007934  {$ENDC}
007935
007936  {$IFC NOT UsingIncludes}
007937   UNIT GestaltEqu;
007938   INTERFACE
007939  {$ENDC}
007940
007941  {$IFC UNDEFINED UsingGestaltEqu}
007942  {$SETC UsingGestaltEqu := 1}
007943
007944  {$I+}
007945  {$SETC GestaltEquIncludes := UsingIncludes}
007946  {$SETC UsingIncludes := 1}
007947  {$IFC UNDEFINED UsingTypes}
007948  {$I $$Shell(PInterfaces)Types.p}
007949  {$ENDC}
007950  {$SETC UsingIncludes := GestaltEquIncludes}
007951
007952  CONST
007953
007954  {**********************
007955   *    Gestalt error codes
007956   **********************}
007957  gestaltUnknownErr = -5550;                    { value returned if Gestalt doesn't know the answer }
007958  gestaltUndefSelectorErr = -5551;              { undefined selector was passed to Gestalt }
007959  gestaltDupSelectorErr = -5552;                { tried to add an entry that already existed }
007960  gestaltLocationErr = -5553;                   { gestalt function ptr wasn't in sysheap }
007961
007962  {************************
007963   *    Environment Selectors
007964   ************************}
007965  gestaltVersion = 'vers';                      { gestalt version }
007966  gestaltAddressingModeAttr = 'addr';           { addressing mode attributes }
007967  gestalt32BitAddressing = 0;                   { using 32-bit addressing mode }
007968  gestalt32BitSysZone = 1;                      { 32-bit compatible system zone }
007969  gestalt32BitCapable = 2;                      { Machine is 32-bit capable }
007970  gestaltAliasMgrAttr = 'alis';                 { Alias Mgr Attributes }
007971  gestaltAliasMgrPresent = 0;                   { True if the Alias Mgr is present }
```

```
007972  gestaltAliasMgrSupportsRemoteAppletalk = 1;     { True if the Alias Mgr knows about Remote Appletalk }
007973  gestaltAppleTalkVersion = 'atlk';               { appletalk version }
007974  gestaltAUXVersion = 'a/ux';                     {a/ux version, if present }
007975  gestaltConnMgrAttr = 'conn';                    { connection mgr attributes }
007976  gestaltConnMgrPresent = 0;
007977  gestaltConnMgrCMSearchFix = 1;                  { Fix to CMAddSearch?  }
007978  gestaltConnMgrErrorString = 2;                  { has CMGetErrorString() }
007979  gestaltConnMgrMultiAsyncIO = 3;                 { CMNewIOPB, CMDisposeIOPB, CMPBRead, CMPBWrite, CMPBIOKill }
007980  gestaltCRMAttr = 'crm ';                        { comm resource mgr attributes }
007981  gestaltCRMPresent = 0;
007982  gestaltCRMPersistentFix = 1;                    { fix for persistent tools }
007983  gestaltCRMToolRsrcCalls = 2;                    { has CRMGetToolResource/ReleaseToolResource }
007984  gestaltCTBVersion = 'ctbv';                     { CommToolbox version }
007985  gestaltDBAccessMgrAttr = 'dbac';                { Database Access Mgr attributes }
007986  gestaltDBAccessMgrPresent = 0;                  { True if Database Access Mgr present }
007987  gestaltDITLExtAttr = 'ditl';                    { AppenDITL, etc. calls from CTB }
007988  gestaltDITLExtPresent = 0;                      { True if calls are present }
007989  gestaltEasyAccessAttr = 'easy';                 { Easy Access attributes }
007990  gestaltEasyAccessOff = 0;                       { if Easy Access present, but off (no icon) }
007991  gestaltEasyAccessOn = 1;                        { if Easy Access "On" }
007992  gestaltEasyAccessSticky = 2;                    { if Easy Access "Sticky" }
007993  gestaltEasyAccessLocked = 3;                    { if Easy Access "Locked" }
007994  gestaltEditionMgrAttr = 'edtn';                 { Edition Mgr attributes }
007995  gestaltEditionMgrPresent = 0;                   { True if Edition Mgr present }
007996  gestaltAppleEventsAttr = 'evnt';                { Apple Events attributes }
007997  gestaltAppleEventsPresent = 0;                  { True if Apple Events present }
007998  gestaltFindFolderAttr = 'fold';                 { Folder Mgr attributes }
007999  gestaltFindFolderPresent = 0;                   { True if Folder Mgr present }
008000  gestaltFontMgrAttr = 'font';                    { Font Mgr attributes }
008001  gestaltOutlineFonts = 0;                        { True if Outline Fonts supported }
008002  gestaltFPUType = 'fpu ';                        { fpu type }
008003  gestaltNoFPU = 0;                               { no FPU }
008004  gestalt68881 = 1;                               { 68881 FPU }
008005  gestalt68882 = 2;                               { 68882 FPU }
008006  gestalt68040FPU = 3;                            { 68040 built-in FPU }
008007  gestaltFSAttr = 'fs  ';                         { file system attributes }
008008  gestaltFullExtFSDispatching = 0;                { has really cool new HFSDispatch dispatcher }
008009  gestaltHasFSSpecCalls = 1;                      { has FSSpec calls }
008010  gestaltHasFileSystemManager = 2;                { has a file system manager }
008011  gestaltFXfrMgrAttr = 'fxfr';                    { file transfer manager attributes }
008012  gestaltFXfrMgrPresent = 0;
008013  gestaltFXfrMgrMultiFile = 1;                    { supports FTSend and FTReceive }
008014  gestaltFXfrMgrErrorString = 2;                  { supports FTGetErrorString }
008015  gestaltHardwareAttr = 'hdwr';                   { hardware attributes }
008016  gestaltHasVIA1 = 0;                             { VIA1 exists }
008017  gestaltHasVIA2 = 1;                             { VIA2 exists }
008018  gestaltHasASC = 3;                              { Apple Sound Chip exists }
008019  gestaltHasSCC = 4;                              { SCC exists }
008020  gestaltHasSCSI = 7;                             { SCSI exists }
008021  gestaltHasSoftPowerOff = 19;                    { Capable of software power off }
008022  gestaltHasSCSI961 = 21;                         { 53C96 SCSI controller on internal bus }
008023  gestaltHasSCSI962 = 22;                         { 53C96 SCSI controller on external bus }
008024  gestaltHasUniversalROM = 24;                    { Do we have a Universal ROM?}
008025  gestaltHelpMgrAttr = 'help';                    { Help Mgr Attributes }
008026  gestaltHelpMgrPresent = 0;                      { true if help mgr is present }
008027  gestaltKeyboardType = 'kbd ';                   { keyboard type }
```

```
008028  gestaltMacKbd = 1;
008029  gestaltMacAndPad = 2;
008030  gestaltMacPlusKbd = 3;
008031  gestaltExtADBKbd = 4;
008032  gestaltStdADBKbd = 5;
008033  gestaltPrtblADBKbd = 6;
008034  gestaltPrtblISOKbd = 7;
008035  gestaltStdISOADBKbd = 8;
008036  gestaltExtISOADBKbd = 9;
008037  gestaltADBKbdII = 10;
008038  gestaltADBISOKbdII = 11;
008039  gestaltPwrBookADBKbd = 12;
008040  gestaltPwrBookISOADBKbd = 13;
008041  gestaltLowMemorySize = 'lmem';                  { size of low memory area }
008042  gestaltLogicalRAMSize = 'lram';                 { logical ram size }
008043  gestaltMiscAttr = 'misc';                       { miscellaneous attributes }
008044  gestaltScrollingThrottle = 0;                   { true if scrolling throttle on }
008045  gestaltSquareMenuBar = 2;                       { true if menu bar is square }
008046  gestaltMMUType = 'mmu ';                        { mmu type }
008047  gestaltNoMMU = 0;                               { no MMU }
008048  gestaltAMU = 1;                                 { address management unit }
008049  gestalt68851 = 2;                               { 68851 PMMU }
008050  gestalt68030MMU = 3;                            { 68030 built-in MMU }
008051  gestalt68040MMU = 4;                            { 68040 built-in MMU }
008052  gestaltStdNBPAttr = 'nlup';                     { standard nbp attributes }
008053  gestaltStdNBPPresent = 0;
008054  gestaltNotificationMgrAttr = 'nmgr';            { notification manager attributes }
008055  gestaltNotificationPresent = 0;                 { notification manager exists }
008056  gestaltNuBusConnectors = 'sltc';                { bitmap of NuBus connectors}
008057  gestaltOSAttr = 'os  ';                         { o/s attributes }
008058  gestaltSysZoneGrowable = 0;                     { system heap is growable }
008059  gestaltLaunchCanReturn = 1;                     { can return from launch }
008060  gestaltLaunchFullFileSpec = 2;                  { can launch from full file spec }
008061  gestaltLaunchControl = 3;                       { launch control support available }
008062  gestaltTempMemSupport = 4;                      { temp memory support }
008063  gestaltRealTempMemory = 5;                      { temp memory handles are real }
008064  gestaltTempMemTracked = 6;                      { temporary memory handles are tracked }
008065  gestaltIPCSupport = 7;                          { IPC support is present }
008066  gestaltSysDebuggerSupport = 8;                  { system debugger support is present }
008067  gestaltOSTable = 'ostt';                        {  OS trap table base  }
008068  gestaltToolboxTable = 'tbtt';                   {  OS trap table base  }
008069  gestaltExtToolboxTable = 'xttt';                { Extended Toolbox trap table base }
008070  gestaltLogicalPageSize = 'pgsz';                { logical page size }
008071  gestaltPowerMgrAttr = 'powr';                   { power manager attributes }
008072  gestaltPMgrExists = 0;
008073  gestaltPMgrCPUIdle = 1;
008074  gestaltPMgrSCC = 2;
008075  gestaltPMgrSound = 3;
008076  gestaltPPCToolboxAttr = 'ppc ';                 { PPC toolbox attributes }
008077
008078  {
008079   * PPC will return the combination of following bit fields.
008080   * e.g. gestaltPPCSupportsRealTime +gestaltPPCSupportsIncoming + gestaltPPCSupportsOutGoing
008081   * indicates PPC is cuurently is only supports real time delivery
008082   * and both incoming and outgoing network sessions are allowed.
008083   * By default local real time delivery is supported as long as PPCInit has been called.}
```

```
008084
008085  gestaltPPCToolboxPresent = $0000;               { PPC Toolbox is present  Requires PPCInit to be called }
008086  gestaltPPCSupportsRealTime = $1000;             { PPC Supports real-time delivery }
008087  gestaltPPCSupportsIncoming = $0001;             { PPC will deny incoming network requests }
008088  gestaltPPCSupportsOutGoing = $0002;             { PPC will deny outgoing network requests }
008089  gestaltProcessorType = 'proc';                  { processor type }
008090  gestalt68000 = 1;
008091  gestalt68010 = 2;
008092  gestalt68020 = 3;
008093  gestalt68030 = 4;
008094  gestalt68040 = 5;
008095  gestaltParityAttr = 'prty';                     { parity attributes }
008096  gestaltHasParityCapability = 0;                 { has ability to check parity }
008097  gestaltParityEnabled = 1;                       { parity checking enabled }
008098  gestaltQuickdrawVersion = 'qd  ';               { quickdraw version }
008099  gestaltOriginalQD = $000;                       { original 1-bit QD }
008100  gestalt8BitQD = $100;                           { 8-bit color QD }
008101  gestalt32BitQD = $200;                          { 32-bit color QD }
008102  gestalt32BitQD11 = $210;                        { 32-bit color QDv1.1 }
008103  gestalt32BitQD12 = $220;                        { 32-bit color QDv1.2 }
008104  gestalt32BitQD13 = $230;                        { 32-bit color QDv1.3 }
008105  gestaltQuickdrawFeatures = 'qdrw';              { quickdraw features }
008106  gestaltHasColor = 0;                            { color quickdraw present }
008107  gestaltHasDeepGWorlds = 1;                      { GWorlds can be deeper than 1-bit }
008108  gestaltHasDirectPixMaps = 2;                    { PixMaps can be direct (16 or 32 bit) }
008109  gestaltHasGrayishTextOr = 3;                    { supports text mode grayishTextOr }
008110  gestaltPhysicalRAMSize = 'ram ';                { physical RAM size }
008111  gestaltPopupAttr = 'pop!';                      { popup cdef attributes }
008112  gestaltPopupPresent = 0;
008113  gestaltResourceMgrAttr = 'rsrc';                { Resource Mgr attributes }
008114  gestaltPartialRsrcs = 0;                        { True if partial resources exist }
008115  gestaltScriptMgrVersion = 'scri';               { Script Manager version number    <08/05/89 pke> }
008116  gestaltScriptCount = 'scr#';                    { number of active script systems    <08/05/89 pke> }
008117  gestaltSerialAttr = 'ser ';                     { Serial attributes }
008118  gestaltHasGPIaToDCDa = 0;                       { GPIa connected to DCDa}
008119  gestaltHasGPIaToRTxCa = 1;                      { GPIa connected to RTxCa clock input}
008120  gestaltHasGPIbToDCDb = 2;                       { GPIb connected to DCDb }
008121  gestaltSoundAttr = 'snd ';                      { sound attributes }
008122  gestaltStereoCapability = 0;                    { sound hardware has stereo capability }
008123  gestaltStereoMixing = 1;                        { stereo mixing on external speaker }
008124  gestaltSoundIOMgrPresent = 3;                   { The Sound I/O Manager is present }
008125  gestaltBuiltInSoundInput = 4;                   { built-in Sound Input hardware is present }
008126  gestaltHasSoundInputDevice = 5;                 { Sound Input device available }
008127  gestaltStandardFileAttr = 'stdf';               { Standard File attributes }
008128  gestaltStandardFile58 = 0;                      { True if selectors 5-8 (StandardPutFile-CustomGetFile) are supported }
008129  gestaltTextEditVersion = 'te  ';                { TextEdit version number            <08/05/89 pke> }
008130  gestaltTE1 = 1;                                 { TextEdit in MacIIci ROM <8Aug89smb> }
008131  gestaltTE2 = 2;                                 { TextEdit with 6.0.4 Script Systems on MacIIci (Script bug fixes for MacIIci) <8Aug89smb> }
008132  gestaltTE3 = 3;                                 { TextEdit with 6.0.4 Script Systems all but MacIIci <8Aug89smb> }
008133  gestaltTE4 = 4;                                 { TextEdit in System 7.0 }
008134  gestaltTE5 = 5;                                 { TextWidthHook available in TextEdit }
008135  gestaltTermMgrAttr = 'term';                    { terminal mgr attributes }
008136  gestaltTermMgrPresent = 0;
008137  gestaltTermMgrErrorString = 2;
008138  gestaltTimeMgrVersion = 'tmgr';                 { time mgr version }
008139  gestaltStandardTimeMgr = 1;                     { standard time mgr is present }
```

```
008140  gestaltRevisedTimeMgr = 2;                      { revised time mgr is present }
008141  gestaltExtendedTimeMgr = 3;                     { extended time mgr is present }
008142  gestaltVMAttr = 'vm  ';                         { virtual memory attributes }
008143  gestaltVMPresent = 0;                           { true if virtual memory is present }
008144
008145  {************************
008146   *    Info-only selectors
008147   ************************}
008148  gestaltMachineType = 'mach';                    { machine type }
008149  kMachineNameStrID = -16395;
008150  gestaltClassic = 1;
008151  gestaltMacXL = 2;
008152  gestaltMac512KE = 3;
008153  gestaltMacPlus = 4;
008154  gestaltMacSE = 5;
008155  gestaltMacII = 6;
008156  gestaltMacIIx = 7;
008157  gestaltMacIIcx = 8;
008158  gestaltMacSE030 = 9;
008159  gestaltPortable = 10;
008160  gestaltMacIIci = 11;
008161  gestaltMacIIfx = 13;
008162  gestaltMacClassic = 17;
008163  gestaltMacIIsi = 18;
008164  gestaltMacLC = 19;
008165  gestaltQuadra900 = 20;
008166  gestaltPowerBook170 = 21;
008167  gestaltQuadra700 = 22;
008168  gestaltClassicII = 23;
008169  gestaltPowerBook100 = 24;
008170  gestaltPowerBook140 = 25;
008171  gestaltMachineIcon = 'micn';                    { machine icon }
008172  gestaltROMSize = 'rom ';                        { rom size }
008173  gestaltROMVersion = 'romv';                     { rom version }
008174  gestaltSystemVersion = 'sysv';                  { system version}
008175
008176  FUNCTION Gestalt(selector: OSType;VAR response: LONGINT): OSErr;
008177  FUNCTION NewGestalt(selector: OSType;gestaltFunction: ProcPtr): OSErr;
008178  FUNCTION ReplaceGestalt(selector: OSType;gestaltFunction: ProcPtr;VAR oldGestaltFunction: ProcPtr): OSErr;
008179
008180
008181  {$ENDC} { UsingGestaltEqu }
008182
008183  {$IFC NOT UsingIncludes}
008184   END.
008185  {$ENDC}
008186
008187
008188  ### END OF FILE GestaltEqu.p
008189
```

```
008190
008191  ############################################################################
008192  ### FILE: Graf3D.p
008193  ############################################################################
008194
008195  {
008196  Created: Monday, January 7, 1991 at 6:00 AM
008197      Graf3D.p
008198      Pascal Interface to the Macintosh Libraries
008199
008200          Copyright Apple Computer, Inc.    1985-1990
008201          All rights reserved
008202  }
008203
008204
008205  {$IFC UNDEFINED UsingIncludes}
008206  {$SETC UsingIncludes := 0}
008207  {$ENDC}
008208
008209  {$IFC NOT UsingIncludes}
008210      UNIT Graf3D;
008211      INTERFACE
008212  {$ENDC}
008213
008214  {$IFC UNDEFINED UsingGraf3D}
008215  {$SETC UsingGraf3D := 1}
008216
008217  {$I+}
008218  {$SETC Graf3DIncludes := UsingIncludes}
008219  {$SETC UsingIncludes := 1}
008220  {$IFC UNDEFINED UsingQuickdraw}
008221  {$I $$Shell(PInterfaces)Quickdraw.p}
008222  {$ENDC}
008223  {$SETC UsingIncludes := Graf3DIncludes}
008224
008225  CONST
008226  radConst = 3754936;
008227
008228  TYPE
008229  Point3D = RECORD
008230      x: Fixed;
008231      y: Fixed;
008232      z: Fixed;
008233      END;
008234
008235  Point2D = RECORD
008236      x: Fixed;
008237      y: Fixed;
008238      END;
008239
008240  XfMatrix = ARRAY [0..3, 0..3] OF Fixed;
008241
008242  Port3DPtr = ^Port3D;
008243  Port3DHandle = ^Port3DPtr;
008244  Port3D = RECORD
008245      grPort: GrafPtr;
```

```
008246      viewRect: Rect;
008247      xLeft: Fixed;
008248      yTop: Fixed;
008249      xRight: Fixed;
008250      yBottom: Fixed;
008251      pen: Point3D;
008252      penPrime: Point3D;
008253      eye: Point3D;
008254      hSize: Fixed;
008255      vSize: Fixed;
008256      hCenter: Fixed;
008257      vCenter: Fixed;
008258      xCotan: Fixed;
008259      yCotan: Fixed;
008260      ident: BOOLEAN;
008261      xForm: XfMatrix;
008262      END;
008263
008264
008265  PROCEDURE InitGrf3d(port: Port3DHandle);
008266  PROCEDURE Open3DPort(port: Port3DPtr);
008267  PROCEDURE SetPort3D(port: Port3DPtr);
008268  PROCEDURE GetPort3D(VAR port: Port3DPtr);
008269  PROCEDURE MoveTo2D(x: Fixed;y: Fixed);
008270  PROCEDURE MoveTo3D(x: Fixed;y: Fixed;z: Fixed);
008271  PROCEDURE LineTo2D(x: Fixed;y: Fixed);
008272  PROCEDURE Move2D(dx: Fixed;dy: Fixed);
008273  PROCEDURE Move3D(dx: Fixed;dy: Fixed;dz: Fixed);
008274  PROCEDURE Line2D(dx: Fixed;dy: Fixed);
008275  PROCEDURE Line3D(dx: Fixed;dy: Fixed;dz: Fixed);
008276  PROCEDURE ViewPort(r: Rect);
008277  PROCEDURE LookAt(left: Fixed;top: Fixed;right: Fixed;bottom: Fixed);
008278  PROCEDURE ViewAngle(angle: Fixed);
008279  PROCEDURE Identity;
008280  PROCEDURE Scale(xFactor: Fixed;yFactor: Fixed;zFactor: Fixed);
008281  PROCEDURE Translate(dx: Fixed;dy: Fixed;dz: Fixed);
008282  PROCEDURE Pitch(xAngle: Fixed);
008283  PROCEDURE Yaw(yAngle: Fixed);
008284  PROCEDURE Roll(zAngle: Fixed);
008285  PROCEDURE Skew(zAngle: Fixed);
008286  PROCEDURE Transform(src: Point3D;VAR dst: Point3D);
008287  FUNCTION Clip3D(src1: Point3D;src2: Point3D;VAR dst1: Point;VAR dst2: Point): INTEGER;
008288  PROCEDURE SetPt3D(VAR pt3D: Point3D;x: Fixed;y: Fixed;z: Fixed);
008289  PROCEDURE SetPt2D(VAR pt2D: Point2D;x: Fixed;y: Fixed);
008290  PROCEDURE LineTo3D(x: Fixed;y: Fixed;z: Fixed);
008291
008292
008293  {$ENDC}    { UsingGraf3D }
008294
008295  {$IFC NOT UsingIncludes}
008296      END.
008297  {$ENDC}
008298
008299
008300  ### END OF FILE Graf3D.p
008301
```

```
008302
008303  ############################################################################
008304  ### FILE: HyperXCmd.p
008305  ############################################################################
008306
008307  {
008308      HyperXCmd.p
008309      Definition file for HyperCard XCMDs and XFCNs in Pascal.
008310
008311      Copyright Apple Computer, Inc.  1987-1991
008312      All rights reserved
008313  }
008314
008315
008316  {$IFC UNDEFINED UsingIncludes}
008317  {$SETC UsingIncludes := 0}
008318  {$ENDC}
008319
008320  {$IFC NOT UsingIncludes}
008321      UNIT HyperXCmd;
008322      INTERFACE
008323  {$ENDC}
008324
008325  {$IFC UNDEFINED UsingHyperXCmd}
008326  {$SETC UsingHyperXCmd := 1}
008327
008328  {$I+}
008329  {$SETC HyperXCmdIncludes := UsingIncludes}
008330  {$SETC UsingIncludes := 1}
008331  {$IFC UNDEFINED UsingTypes}
008332  {$I $$Shell(PInterfaces)Types.p}
008333  {$ENDC}
008334  {$IFC UNDEFINED UsingEvents}
008335  {$I $$Shell(PInterfaces)Events.p}
008336  {$ENDC}
008337  {$IFC UNDEFINED UsingTextEdit}
008338  {$I $$Shell(PInterfaces)TextEdit.p}
008339  {$ENDC}
008340  {$IFC UNDEFINED UsingMenus}
008341  {$I $$Shell(PInterfaces)Menus.p}
008342  {$ENDC}
008343  {$IFC UNDEFINED UsingStandardFile}
008344  {$I $$Shell(PInterfaces)StandardFile.p}
008345  {$ENDC}
008346  {$SETC UsingIncludes := HyperXCmdIncludes}
008347
008348  CONST
008349    { result codes }
008350    xresSucc = 0;
008351    xresFail = 1;
008352    xresNotImp = 2;
008353
008354    { XCMDBlock constants for event.what... }
008355    xOpenEvt              = 1000;         { the first event after you are created }
008356    xCloseEvt             = 1001;         { your window is being forced close (Quit?) }
008357    xGiveUpEditEvt        = 1002;         { you are losing Edit... }
```

```
008358    xGiveUpSoundEvt   = 1003;     { someone else is requesting HyperCard's sound channel }
008359    xHidePalettesEvt  = 1004;     { someone called HideHCPalettes }
008360    xShowPalettesEvt  = 1005;     { someone called ShowHCPalettes }
008361    xEditUndo     = 1100;     { Edit——Undo }
008362    xEditCut      = 1102;     { Edit——Cut }
008363    xEditCopy     = 1103;     { Edit——Copy }
008364    xEditPaste        = 1104;     { Edit——Paste }
008365    xEditClear        = 1105;     { Edit——Clear }
008366    xSendEvt      = 1200;     { script has sent you a message (text) }
008367    xSetPropEvt       = 1201;     { set a window property }
008368    xGetPropEvt       = 1202;     { get a window property }
008369    xCursorWithin     = 1300;     { cursor is within the window }
008370    xMenuEvt      = 1400;     { user has selected an item in your menu }
008371    xMBarClickedEvt   = 1401;     { a menu is about to be shown--update if needed }
008372
008373    xShowWatchInfoEvt = 1501;     { for variable and message watchers }
008374    xScriptErrorEvt   = 1502;     { place the insertion point }
008375    xDebugErrorEvt    = 1503;     { user clicked "Debug" at a complaint }
008376    xDebugStepEvt     = 1504;     { hilite the line }
008377    xDebugTraceEvt    = 1505;     { same as step but tracing }
008378    xDebugFinishedEvt = 1506;     { script ended }
008379
008380    paletteProc       = 2048;     { Windoid with grow box }
008381    palNoGrowProc     = 2052;     { standard Windoid defproc }
008382    palZoomProc       = 2056;     { Windoid with zoom and grow }
008383    palZoomNoGrow     = 2060;     { Windoid with zoom and no grow }
008384    hasZoom       =    8;
008385    hasTallTBar       =    2;
008386    toggleHilite      =    1;
008387
008388    maxCachedChecks   = 16;       { maximum number of checkpoints in a script }
008389
008390    { paramCount is set to these constants when first calling special XThings }
008391    xMessageWatcherID = -2;
008392    xVariableWatcherID    = -3;
008393    xScriptEditorID   = -4;
008394    xDebuggerID       = -5;
008395
008396    { XTalkObjectPtr^.objectKind values }
008397    stackObj  = 1;
008398    bkgndObj  = 2;
008399    cardObj   = 3;
008400    fieldObj  = 4;
008401    buttonObj = 5;
008402
008403    { selectors for ShowHCAlert's dialogs (shown as buttonID:buttonText) }
008404    errorDlgID        = 1;    { 1:OK (default) }
008405    confirmDlgID      = 2;    { 1:OK (default) and 2:Cancel }
008406    confirmDelDlgID   = 3;    { 1:Cancel (default) and 2:Delete }
008407    yesNoCancelDlgID  = 4;    { 1:Yes (default), 2:Cancel, and 3:No }
008408
008409
008410  TYPE
008411    XCmdPtr   = ^XCmdBlock;
008412    XCmdBlock = RECORD
008413          paramCount: INTEGER; { If = -1 then new use for XWindoids }
```

```
008414          params:     ARRAY[1..16] OF Handle;
008415          returnValue:    Handle;
008416          passFlag:   BOOLEAN;
008417
008418          entryPoint: ProcPtr; { to call back to HyperCard }
008419          request:    INTEGER;
008420          result:     INTEGER;
008421          inArgs:     ARRAY[1..8] OF LongInt;
008422          outArgs:    ARRAY[1..4] OF LongInt;
008423            END;
008424
008425    XWEventInfoPtr = ^XWEventInfo;
008426    XWEventInfo    = RECORD
008427                event:   EventRecord;
008428                eventWindow: WindowPtr;
008429                eventParams: ARRAY[1..9] OF LongInt;
008430                eventResult: Handle;
008431             END;
008432
008433    XTalkObjectPtr = ^XTalkObject;
008434    XTalkObject = RECORD
008435            objectKind:   INTEGER;    { stack, bkgnd, card, field, or button }
008436            stackNum: LongInt;    { reference number of the source stack }
008437            bkgndID:  LongInt;
008438            cardID:   LongInt;
008439            buttonID: LongInt;
008440            fieldID:  LongInt;
008441          END;
008442
008443    CheckPtHandle = ^CheckPtPtr;
008444    CheckPtPtr    = ^CheckPts;
008445    CheckPts  = RECORD
008446                        checks: ARRAY[1..maxCachedChecks] OF INTEGER;
008447            END;
008448
008449
008450  (****  HyperTalk Utilities  ****)
008451  FUNCTION  EvalExpr(paramPtr: XCmdPtr; expr: Str255): Handle;
008452  PROCEDURE SendCardMessage(paramPtr: XCmdPtr; msg: Str255);
008453  PROCEDURE SendHCMessage(paramPtr: XCmdPtr; msg: Str255);
008454  PROCEDURE RunHandler(paramPtr: XCmdPtr; handler: Handle);
008455
008456  (****  Memory Utilities  ****)
008457  FUNCTION  GetGlobal(paramPtr: XCmdPtr; globName: Str255): Handle;
008458  PROCEDURE SetGlobal(paramPtr: XCmdPtr; globName: Str255; globValue: Handle);
008459  PROCEDURE ZeroBytes(paramPtr: XCmdPtr; dstPtr: Ptr;longCount: LongInt);
008460
008461  (****  String Utilities  ****)
008462  PROCEDURE ScanToReturn(paramPtr: XCmdPtr; VAR scanPtr: Ptr);
008463  PROCEDURE ScanToZero(paramPtr: XCmdPtr; VAR scanPtr: Ptr);
008464  FUNCTION  StringEqual(paramPtr: XCmdPtr; str1,str2: Str255): BOOLEAN;
008465  FUNCTION  StringLength(paramPtr: XCmdPtr; strPtr: Ptr): LongInt;
008466  FUNCTION  StringMatch(paramPtr: XCmdPtr; pattern: Str255; target: Ptr): Ptr;
008467  PROCEDURE ZeroTermHandle(paramPtr: XCmdPtr; hndl: Handle);
008468
008469  (****  String Conversions  ****)
```

```
008470  PROCEDURE BoolToStr(paramPtr: XCmdPtr; bool: BOOLEAN; VAR str: Str255);
008471  PROCEDURE ExtToStr(paramPtr: XCmdPtr; num: Extended80; VAR str: Str255);
008472  PROCEDURE LongToStr(paramPtr: XCmdPtr; posNum: LongInt; VAR str: Str255);
008473  PROCEDURE NumToHex(paramPtr: XCmdPtr; num: LongInt; nDigits: INTEGER; VAR str: Str255);
008474  PROCEDURE NumToStr(paramPtr: XCmdPtr; num: LongInt; VAR str: Str255);
008475  FUNCTION  PasToZero(paramPtr: XCmdPtr; str: Str255): Handle;
008476  PROCEDURE PointToStr(paramPtr: XCmdPtr; pt: Point; VAR str: Str255);
008477  PROCEDURE RectToStr(paramPtr: XCmdPtr; rct: Rect; VAR str: Str255);
008478  PROCEDURE ReturnToPas(paramPtr: XCmdPtr; zeroStr: Ptr; VAR pasStr: Str255);
008479  FUNCTION  StrToBool(paramPtr: XCmdPtr; str: Str255): BOOLEAN;
008480  FUNCTION  StrToExt(paramPtr: XCmdPtr; str: Str255): Extended80;
008481  FUNCTION  StrToLong(paramPtr: XCmdPtr; str: Str255): LongInt;
008482  FUNCTION  StrToNum(paramPtr: XCmdPtr; str: Str255): LongInt;
008483  PROCEDURE StrToPoint(paramPtr: XCmdPtr; str: Str255; VAR pt: Point);
008484  PROCEDURE StrToRect(paramPtr: XCmdPtr; str: Str255; VAR rct: Rect);
008485  PROCEDURE ZeroToPas(paramPtr: XCmdPtr; zeroStr: Ptr; VAR pasStr: Str255);
008486
008487  (****  Field Utilities  ****)
008488  FUNCTION  GetFieldByID(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldID: INTEGER): Handle;
008489  FUNCTION  GetFieldByName(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldName: Str255): Handle;
008490  FUNCTION  GetFieldByNum(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldNum: INTEGER): Handle;
008491  PROCEDURE SetFieldByID(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldID: INTEGER; fieldVal: Handle);
008492  PROCEDURE SetFieldByName(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldName: Str255; fieldVal: Handle);
008493  PROCEDURE SetFieldByNum(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldNum: INTEGER; fieldVal: Handle);
008494  FUNCTION  GetFieldTE(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldID,fieldNum: INTEGER;
008495              fieldNamePtr: StringPtr): TEHandle;
008496  PROCEDURE SetFieldTE(paramPtr: XCmdPtr; cardFieldFlag: BOOLEAN; fieldID,fieldNum: INTEGER;
008497              fieldNamePtr: StringPtr; fieldTE: TEHandle);
008498
008499  (****  Miscellaneous Utilities  ****)
008500  PROCEDURE BeginXSound(paramPtr: XCmdPtr; window: WindowPtr);
008501  PROCEDURE EndXSound(paramPtr: XCmdPtr);
008502  FUNCTION  GetFilePath(paramPtr: XCmdPtr; fileName: Str255; numTypes: INTEGER; typeList: SFTypeList;
008503               askUser: BOOLEAN; VAR fileType: OSType; VAR fullName: Str255): BOOLEAN;
008504  PROCEDURE GetXResInfo(paramPtr: XCmdPtr; VAR resFile: INTEGER; VAR resID: INTEGER;
008505               VAR rType: ResType; VAR name: Str255);
008506  PROCEDURE Notify(paramPtr: XCmdPtr);
008507  PROCEDURE SendHCEvent(paramPtr: XCmdPtr; event: EventRecord);
008508  PROCEDURE SendWindowMessage(paramPtr: XCmdPtr; windPtr: WindowPtr;
008509                 windowName: Str255; msg: Str255);
008510  FUNCTION  FrontDocWindow(paramPtr: XCmdPtr): WindowPtr;
008511  FUNCTION  StackNameToNum(paramPtr: XCmdPtr; stackName: Str255): LongInt;
008512  FUNCTION  ShowHCAlert(paramPtr: XCMDPtr; dlgID: INTEGER; promptStr: Str255): INTEGER;
008513
008514  (****  Creating and Disposing XWindoids  ****)
008515  FUNCTION  NewXWindow(paramPtr: XCmdPtr; boundsRect: Rect; title: Str255; visible: BOOLEAN;
008516              procID: INTEGER; color: BOOLEAN; floating: BOOLEAN): WindowPtr;
008517  FUNCTION  GetNewXWindow(paramPtr: XCmdPtr; templateType: ResType; templateID: INTEGER;
008518             color: BOOLEAN; floating: BOOLEAN): WindowPtr;
008519  PROCEDURE CloseXWindow(paramPtr: XCmdPtr; window: WindowPtr);
008520
008521  (****  XWindoid Utilities  ****)
008522  PROCEDURE HideHCPalettes(paramPtr: XCmdPtr);
008523  PROCEDURE ShowHCPalettes(paramPtr: XCmdPtr);
008524  PROCEDURE RegisterXWMenu(paramPtr: XCmdPtr; window: WindowPtr; menu: MenuHandle; registering: BOOLEAN);
008525  PROCEDURE SetXWIdleTime(paramPtr: XCmdPtr; window: WindowPtr; interval: LongInt);
```

```
008526  PROCEDURE XWHasInterruptCode(paramPtr: XCmdPtr; window: WindowPtr; haveCode: BOOLEAN);
008527  PROCEDURE XWAlwaysMoveHigh(paramPtr: XCmdPtr; window: WindowPtr; moveHigh: BOOLEAN);
008528  PROCEDURE XWAllowReEntrancy(paramPtr: XCmdPtr; window: WindowPtr; allowSysEvts: BOOLEAN; allowHCEvts: BOOLEAN);
008529
008530  (****  Text Editing Utilities  ****)
008531  PROCEDURE BeginXWEdit(paramPtr: XCmdPtr; window: WindowPtr);
008532  PROCEDURE EndXWEdit(paramPtr: XCmdPtr; window: WindowPtr);
008533  FUNCTION  HCWordBreakProc(paramPtr: XCmdPtr): ProcPtr;
008534  PROCEDURE PrintTEHandle(paramPtr: XCmdPtr; hTE: TEHandle; header: StringPtr);
008535
008536  (****  Script Editor support  ****)
008537  FUNCTION  GetCheckPoints(paramPtr: XCmdPtr): CheckPtHandle;
008538  PROCEDURE SetCheckPoints(paramPtr: XCmdPtr; checkLines: CheckPtHandle);
008539  PROCEDURE FormatScript(paramPtr: XCmdPtr; scriptHndl: Handle;
008540                VAR insertionPoint: LongInt; quickFormat: BOOLEAN);
008541  PROCEDURE SaveXWScript(paramPtr: XCmdPtr; scriptHndl: Handle);
008542  PROCEDURE GetObjectName(paramPtr: XCmdPtr; object: XTalkObjectPtr; VAR objName: Str255);
008543  PROCEDURE GetObjectScript(paramPtr: XCmdPtr; object: XTalkObjectPtr; VAR scriptHndl: Handle);
008544  PROCEDURE SetObjectScript(paramPtr: XCmdPtr; object: XTalkObjectPtr; scriptHndl: Handle);
008545
008546  (****  Debugging Tools support  ****)
008547  PROCEDURE AbortScript(paramPtr: XCmdPtr);
008548  PROCEDURE GoScript(paramPtr: XCmdPtr);
008549  PROCEDURE StepScript(paramPtr: XCmdPtr; stepInto: BOOLEAN);
008550  PROCEDURE CountHandlers(paramPtr: XCmdPtr; VAR handlerCount: INTEGER);
008551  PROCEDURE GetHandlerInfo(paramPtr: XCmdPtr; handlerNum: INTEGER; VAR handlerName: Str255;
008552              VAR objectName: Str255; VAR varCount: INTEGER);
008553  PROCEDURE GetVarInfo(paramPtr: XCmdPtr; handlerNum: INTEGER; varNum: INTEGER;
008554              VAR varName: Str255; VAR isGlobal: BOOLEAN; VAR varValue: Str255;
008555              varHndl: Handle);
008556  PROCEDURE SetVarValue(paramPtr: XCmdPtr; handlerNum: INTEGER; varNum: INTEGER;
008557               varHndl: Handle);
008558  FUNCTION  GetStackCrawl(paramPtr: XCmdPtr): Handle;
008559  PROCEDURE TraceScript(paramPtr: XCmdPtr; traceInto: BOOLEAN);
008560
008561
008562  {$ENDC}    { UsingHyperXCmd }
008563
008564  {$IFC NOT UsingIncludes}
008565      END.
008566  {$ENDC}
008567
008568  ### END OF FILE HyperXCmd.p
008569
```

```
008570
008571  ############################################################################
008572  ### FILE: Icons.p
008573  ############################################################################
008574
008575
008576  {
008577  Created: Tuesday, September 10, 1991 at 2:03 PM
008578   Icons.p
008579   Pascal Interface to the Macintosh Libraries
008580
008581    Copyright Apple Computer, Inc. 1990-1991
008582    All rights reserved
008583  }
008584
008585
008586  {$IFC UNDEFINED UsingIncludes}
008587  {$SETC UsingIncludes := 0}
008588  {$ENDC}
008589
008590  {$IFC NOT UsingIncludes}
008591   UNIT Icons;
008592   INTERFACE
008593  {$ENDC}
008594
008595  {$IFC UNDEFINED UsingIcons}
008596  {$SETC UsingIcons := 1}
008597
008598
008599  CONST
008600
008601  { The following are icons for which there are both icon suites and SICNs. }
008602  genericDocumentIconResource = -4000;
008603  genericStationeryIconResource = -3985;
008604  genericEditionFileIconResource = -3989;
008605  genericApplicationIconResource = -3996;
008606  genericDeskAccessoryIconResource = -3991;
008607
008608  genericFolderIconResource = -3999;
008609  privateFolderIconResource = -3994;
008610
008611  floppyIconResource = -3998;
008612  trashIconResource = -3993;
008613
008614  { The following are icons for which there are SICNs only. }
008615  desktopIconResource = -3992;
008616  openFolderIconResource = -3997;
008617  genericHardDiskIconResource = -3995;
008618  genericFileServerIconResource = -3972;
008619  genericSuitcaseIconResource = -3970;
008620  genericMoverObjectIconResource = -3969;
008621
008622  { The following are icons for which there are icon suites only. }
008623  genericPreferencesIconResource = -3971;
008624  genericQueryDocumentIconResource = -16506;
008625  genericExtensionIconResource = -16415;
```

```
008626
008627  systemFolderIconResource = -3983;
008628  appleMenuFolderIconResource = -3982;
008629  startupFolderIconResource = -3981;
008630  ownedFolderIconResource = -3980;
008631  dropFolderIconResource = -3979;
008632  sharedFolderIconResource = -3978;
008633  mountedFolderIconResource = -3977;
008634  controlPanelFolderIconResource = -3976;
008635  printMonitorFolderIconResource = -3975;
008636  preferencesFolderIconResource = -3974;
008637  extensionsFolderIconResource = -3973;
008638
008639  fullTrashIconResource = -3984;
008640
008641  large1BitMask = 'ICN#';
008642  large4BitData = 'icl4';
008643  large8BitData = 'icl8';
008644  small1BitMask = 'ics#';
008645  small4BitData = 'ics4';
008646  small8BitData = 'ics8';
008647  mini1BitMask = 'icm#';
008648  mini4BitData = 'icm4';
008649  mini8BitData = 'icm8';
008650
008651
008652  {$ENDC} { UsingIcons }
008653
008654  {$IFC NOT UsingIncludes}
008655   END.
008656  {$ENDC}
008657
008658
008659  ### END OF FILE Icons.p
008660
```

```
008661
008662  ############################################################################
008663  ### FILE: IntEnv.p
008664  ############################################################################
008665
008666  {
008667  Created: Wednesday, June 27, 1990 at 6:42 PM
008668      IntEnv.p
008669      Pascal Interface to the Macintosh Libraries
008670
008671          Copyright Apple Computer, Inc. 1989-1991
008672          All rights reserved
008673  }
008674
008675
008676  {$IFC UNDEFINED UsingIncludes}
008677  {$SETC UsingIncludes := 0}
008678  {$ENDC}
008679
008680  {$IFC NOT UsingIncludes}
008681      UNIT IntEnv;
008682      INTERFACE
008683  {$ENDC}
008684
008685  {$IFC UNDEFINED UsingIntEnv}
008686  {$SETC UsingIntEnv := 1}
008687
008688  {$I+}
008689  {$SETC IntEnvIncludes := UsingIncludes}
008690  {$SETC UsingIncludes := 1}
008691  {$IFC UNDEFINED UsingPasLibIntf}
008692  {$I $$Shell(PInterfaces)PasLibIntf.p}
008693  {$ENDC}
008694  {$SETC UsingIncludes := IntEnvIncludes}
008695
008696      CONST
008697
008698        { CMD words for IEfaccess(), from <fcntl.h> }
008699
008700        F_OPEN = $6400; { (('d'<<8)|00), d => "directory" ops }
008701        F_DELETE = $6401;
008702        F_RENAME = $6402;
008703
008704        F_GTABINFO = $6500; { (('e'<<8)|00), e => "editor" ops }
008705        F_STABINFO = $6501;
008706        F_GFONTINFO = $6502;
008707        F_SFONTINFO = $6503;
008708        F_GPRINTREC = $6504;
008709        F_SPRINTREC = $6505;
008710        F_GSELINFO = $6506;
008711        F_SSELINFO = $6507;
008712        F_GWININFO = $6508;
008713        F_SWININFO = $6509;
008714        F_GSCROLLINFO = $6510;
008715        F_SSCROLLINFO = $6511;
008716
```

```
008717        { Open modes for IEopen(), from <fcntl.h> }
008718
008719        O_RDONLY = $0000;
008720        O_WRONLY = $0001;
008721        O_RDWR = $0002;
008722        O_APPEND = $0008;
008723        O_RSRC = $0010;
008724        O_CREAT = $0100;
008725        O_TRUNC = $0200;
008726        O_EXCL = $0400;
008727
008728        { IOCtl parameters }
008729
008730        FIOINTERACTIVE = $6602; { (('f'<<8)|02), f => "open file" ops }
008731        FIOBUFSIZE = $6603;
008732        FIOFNAME = $6604;
008733        FIOREFNUM = $6605;
008734        FIOSETEOF = $6606;
008735
008736      TYPE
008737        IEString = STRING;
008738        IEStringPtr = ^IEString;
008739        IEStringVec = ARRAY [0..8191] OF IEStringPtr;
008740        IEStringVecPtr = ^IEStringVec;
008741
008742        {$PUSH}
008743        {$J+} { EXPORTed unit globals }
008744
008745      VAR
008746        ArgC: LONGINT;
008747        ArgV: IEStringVecPtr;
008748        _EnvP: IEStringVecPtr;
008749
008750        Diagnostic: TEXT;
008751        {$POP}
008752
008753      FUNCTION IEStandAlone: BOOLEAN;
008754
008755      FUNCTION IEgetenv(envName: STRING; VAR envValue: UNIV IEString): BOOLEAN;
008756
008757      FUNCTION IEfaccess(fName: STRING; opCode: LONGINT;
008758                         arg: UNIV LONGINT): LONGINT;
008759
008760      PROCEDURE IEopen(VAR fvar: UNIV PASCALFILE; fName: STRING; mode: LONGINT);
008761
008762      FUNCTION IEioctl(VAR fvar: UNIV PASCALFILE; request: LONGINT;
008763                       arg: UNIV LONGINT): LONGINT;
008764
008765      FUNCTION IElseek(VAR fvar: UNIV PASCALFILE; offset: LONGINT;
008766                       whence: LONGINT): LONGINT;
008767
008768      PROCEDURE IEatexit(exitProc: UNIV LONGINT);
008769        C;
008770
008771      PROCEDURE IEexit(status: LONGINT);
008772        C;
```

```
008773
008774      PROCEDURE IE_exit(status: LONGINT);
008775        C;
008776
008777  {$ENDC}    { UsingIntEnv }
008778
008779  {$IFC NOT UsingIncludes}
008780      END.
008781  {$ENDC}
008782
008783
008784  ### END OF FILE IntEnv.p
008785
```

```
008786
008787  ############################################################################
008788  ### FILE: Language.p
008789  ############################################################################
008790
008791
008792  {
008793  Created: Sunday, September 15, 1991 at 11:20 PM
008794   Language.p
008795   Pascal Interface to the Macintosh Libraries
008796
008797    Copyright Apple Computer, Inc. 1986-1991
008798    All rights reserved
008799  }
008800
008801
008802  {$IFC UNDEFINED UsingIncludes}
008803  {$SETC UsingIncludes := 0}
008804  {$ENDC}
008805
008806  {$IFC NOT UsingIncludes}
008807   UNIT Language;
008808   INTERFACE
008809  {$ENDC}
008810
008811  {$IFC UNDEFINED UsingLanguage}
008812  {$SETC UsingLanguage := 1}
008813
008814
008815  CONST
008816
008817  { Language Codes }
008818  langEnglish = 0;            { smRoman script }
008819  langFrench = 1;             { smRoman script }
008820  langGerman = 2;             { smRoman script }
008821  langItalian = 3;            { smRoman script }
008822  langDutch = 4;              { smRoman script }
008823  langSwedish = 5;            { smRoman script }
008824  langSpanish = 6;            { smRoman script }
008825  langDanish = 7;             { smRoman script }
008826  langPortuguese = 8;         { smRoman script }
008827  langNorwegian = 9;          { smRoman script }
008828  langHebrew = 10;            { smHebrew script }
008829  langJapanese = 11;          { smJapanese script }
008830  langArabic = 12;            { smArabic script }
008831  langFinnish = 13;           { smRoman script }
008832  langGreek = 14;             { smGreek script }
008833  langIcelandic = 15;         { extended Roman script }
008834  langMaltese = 16;           { extended Roman script }
008835  langTurkish = 17;           { extended Roman script }
008836  langCroatian = 18;          { Serbo-Croatian in extended Roman script }
008837  langTradChinese = 19;       { Chinese in traditional characters }
008838  langUrdu = 20;              { smArabic script }
008839  langHindi = 21;             { smDevanagari script }
008840  langThai = 22;              { smThai script }
008841  langKorean = 23;            { smKorean script }
```

```
008842  langLithuanian = 24;        { smEastEurRoman script }
008843  langPolish = 25;            { smEastEurRoman script }
008844  langHungarian = 26;         { smEastEurRoman script }
008845  langEstonian = 27;          { smEastEurRoman script }
008846  langLettish = 28;           { smEastEurRoman script }
008847  langLatvian = 28;           { Synonym for langLettish }
008848  langLapponian = 29;         { extended Roman script }
008849  langLappish = 29;           { Synonym for langLapponian }
008850  langFaeroese = 30;          { smRoman script }
008851  langFarsi = 31;             { smArabic script }
008852  langPersian = 31;           { Synonym for langFarsi }
008853  langRussian = 32;           { smCyrillic script }
008854  langSimpChinese = 33;       { Chinese in simplified characters }
008855  langFlemish = 34;           { smRoman script }
008856  langIrish = 35;             { smRoman script }
008857  langAlbanian = 36;          { smRoman script }
008858  langRomanian = 37;          { smEastEurRoman script }
008859  langCzech = 38;             { smEastEurRoman script }
008860  langSlovak = 39;            { smEastEurRoman script }
008861  langSlovenian = 40;         { smEastEurRoman script }
008862  langYiddish = 41;           { smHebrew script }
008863  langSerbian = 42;           { Serbo-Croatian in smCyrillic script }
008864  langMacedonian = 43;        { smCyrillic script }
008865  langBulgarian = 44;         { smCyrillic script }
008866  langUkrainian = 45;         { smCyrillic script }
008867  langByelorussian = 46;      { smCyrillic script }
008868  langUzbek = 47;             { smCyrillic script }
008869  langKazakh = 48;            { smCyrillic script }
008870  langAzerbaijani = 49;       { Azerbaijani in smCyrillic script (USSR) }
008871  langAzerbaijanAr = 50;      { Azerbaijani in smArabic script (Iran) }
008872  langArmenian = 51;          { smArmenian script }
008873  langGeorgian = 52;          { smGeorgian script }
008874  langMoldavian = 53;         { smCyrillic script }
008875  langKirghiz = 54;           { smCyrillic script }
008876  langTajiki = 55;            { smCyrillic script }
008877  langTurkmen = 56;           { smCyrillic script }
008878  langMongolian = 57;         { Mongolian in smMongolian script }
008879  langMongolianCyr = 58;      { Mongolian in smCyrillic script }
008880  langPashto = 59;            { smArabic script }
008881  langKurdish = 60;           { smArabic script }
008882  langKashmiri = 61;          { smArabic script }
008883  langSindhi = 62;            { smExtArabic script }
008884  langTibetan = 63;           { smTibetan script }
008885  langNepali = 64;            { smDevanagari script }
008886  langSanskrit = 65;          { smDevanagari script }
008887  langMarathi = 66;           { smDevanagari script }
008888  langBengali = 67;           { smBengali script }
008889  langAssamese = 68;          { smBengali script }
008890  langGujarati = 69;          { smGujarati script }
008891  langPunjabi = 70;           { smGurmukhi script }
008892  langOriya = 71;             { smOriya script }
008893  langMalayalam = 72;         { smMalayalam script }
008894  langKannada = 73;           { smKannada script }
008895  langTamil = 74;             { smTamil script }
008896  langTelugu = 75;            { smTelugu script }
008897  langSinhalese = 76;         { smSinhalese script }
```

```
008898  langBurmese = 77;           { smBurmese script }
008899  langKhmer = 78;             { smKhmer script }
008900  langLao = 79;               { smLaotian script }
008901  langVietnamese = 80;        { smVietnamese script }
008902  langIndonesian = 81;        { smRoman script }
008903  langTagalog = 82;           { smRoman script }
008904  langMalayRoman = 83;        { Malay in smRoman script }
008905  langMalayArabic = 84;       { Malay in smArabic script }
008906  langAmharic = 85;           { smEthiopic script }
008907  langTigrinya = 86;          { smEthiopic script }
008908  langGalla = 87;             { smEthiopic script }
008909  langOromo = 87;             { Synonym for langGalla }
008910  langSomali = 88;            { smRoman script }
008911  langSwahili = 89;           { smRoman script }
008912  langRuanda = 90;            { smRoman script }
008913  langRundi = 91;             { smRoman script }
008914  langChewa = 92;             { smRoman script }
008915  langMalagasy = 93;          { smRoman script }
008916  langEsperanto = 94;         { extended Roman script }
008917  langWelsh = 128;            { smRoman script }
008918  langBasque = 129;           { smRoman script }
008919  langCatalan = 130;          { smRoman script }
008920  langLatin = 131;            { smRoman script }
008921  langQuechua = 132;          { smRoman script }
008922  langGuarani = 133;          { smRoman script }
008923  langAymara = 134;           { smRoman script }
008924  langTatar = 135;            { smCyrillic script }
008925  langUighur = 136;           { smArabic script }
008926  langDzongkha = 137;         { (lang of Bhutan) smTibetan script }
008927  langJavaneseRom = 138;      { Javanese in smRoman script }
008928  langSundaneseRom = 139;     { Sundanese in smRoman script }
008929
008930  { Obsolete names, kept for backward compatibility }
008931  langPortugese = 8;          { old misspelled version, kept for compatibility }
008932  langMalta = 16;             { old misspelled version, kept for compatibility }
008933  langYugoslavian = 18;       { (use langCroatian, langSerbian, etc.) }
008934  langChinese = 19;           { (use langTradChinese or langSimpChinese) }
008935
008936
008937  {$ENDC} { UsingLanguage }
008938
008939  {$IFC NOT UsingIncludes}
008940   END.
008941  {$ENDC}
008942
008943
008944  ### END OF FILE Language.p
008945
```

```
008946
008947  ############################################################################
008948  ### FILE: Lists.p
008949  ############################################################################
008950
008951  {
008952  Created: Monday, January 7, 1991 at 5:54 AM
008953      Lists.p
008954      Pascal Interface to the Macintosh Libraries
008955
008956          Copyright Apple Computer, Inc.    1985-1990
008957          All rights reserved
008958  }
008959
008960
008961  {$IFC UNDEFINED UsingIncludes}
008962  {$SETC UsingIncludes := 0}
008963  {$ENDC}
008964
008965  {$IFC NOT UsingIncludes}
008966      UNIT Lists;
008967      INTERFACE
008968  {$ENDC}
008969
008970  {$IFC UNDEFINED UsingLists}
008971  {$SETC UsingLists := 1}
008972
008973  {$I+}
008974  {$SETC ListsIncludes := UsingIncludes}
008975  {$SETC UsingIncludes := 1}
008976  {$IFC UNDEFINED UsingTypes}
008977  {$I $$Shell(PInterfaces)Types.p}
008978  {$ENDC}
008979  {$IFC UNDEFINED UsingControls}
008980  {$I $$Shell(PInterfaces)Controls.p}
008981  {$ENDC}
008982  {$IFC UNDEFINED UsingMemory}
008983  {$I $$Shell(PInterfaces)Memory.p}
008984  {$ENDC}
008985  {$SETC UsingIncludes := ListsIncludes}
008986
008987  CONST
008988  lDoVAutoscroll = 2;
008989  lDoHAutoscroll = 1;
008990  lOnlyOne = -128;
008991  lExtendDrag = 64;
008992  lNoDisjoint = 32;
008993  lNoExtend = 16;
008994  lNoRect = 8;
008995  lUseSense = 4;
008996  lNoNilHilite = 2;
008997  lInitMsg = 0;
008998  lDrawMsg = 1;
008999  lHiliteMsg = 2;
009000  lCloseMsg = 3;
009001
```

```
009002  TYPE
009003  Cell = Point;
009004
009005  DataPtr = ^DataArray;
009006  DataHandle = ^DataPtr;
009007
009008  DataArray = PACKED ARRAY [0..32000] OF CHAR;
009009
009010  ListPtr = ^ListRec;
009011  ListHandle = ^ListPtr;
009012  ListRec = RECORD
009013      rView: Rect;
009014      port: GrafPtr;
009015      indent: Point;
009016      cellSize: Point;
009017      visible: Rect;
009018      vScroll: ControlHandle;
009019      hScroll: ControlHandle;
009020      selFlags: SignedByte;
009021      lActive: BOOLEAN;
009022      lReserved: SignedByte;
009023      listFlags: SignedByte;
009024      clikTime: LONGINT;
009025      clikLoc: Point;
009026      mouseLoc: Point;
009027      lClikLoop: ProcPtr;
009028      lastClick: Cell;
009029      refCon: LONGINT;
009030      listDefProc: Handle;
009031      userHandle: Handle;
009032      dataBounds: Rect;
009033      cells: DataHandle;
009034      maxIndex: INTEGER;
009035      cellArray: ARRAY [1..1] OF INTEGER;
009036      END;
009037
009038
009039  FUNCTION LNew(rView: Rect;dataBounds: Rect;cSize: Point;theProc: INTEGER;
009040      theWindow: WindowPtr;drawIt: BOOLEAN;hasGrow: BOOLEAN;scrollHoriz: BOOLEAN;
009041      scrollVert: BOOLEAN): ListHandle;
009042      INLINE $3F3C,$0044,$A9E7;
009043  PROCEDURE LDispose(lHandle: ListHandle);
009044      INLINE $3F3C,$0028,$A9E7;
009045  FUNCTION LAddColumn(count: INTEGER;colNum: INTEGER;lHandle: ListHandle): INTEGER;
009046      INLINE $3F3C,$0004,$A9E7;
009047  FUNCTION LAddRow(count: INTEGER;rowNum: INTEGER;lHandle: ListHandle): INTEGER;
009048      INLINE $3F3C,$0008,$A9E7;
009049  PROCEDURE LDelColumn(count: INTEGER;colNum: INTEGER;lHandle: ListHandle);
009050      INLINE $3F3C,$0020,$A9E7;
009051  PROCEDURE LDelRow(count: INTEGER;rowNum: INTEGER;lHandle: ListHandle);
009052      INLINE $3F3C,$0024,$A9E7;
009053  FUNCTION LGetSelect(next: BOOLEAN;VAR theCell: Cell;lHandle: ListHandle): BOOLEAN;
009054      INLINE $3F3C,$003C,$A9E7;
009055  FUNCTION LLastClick(lHandle: ListHandle): Cell;
009056      INLINE $3F3C,$0040,$A9E7;
009057  FUNCTION LNextCell(hNext: BOOLEAN;vNext: BOOLEAN;VAR theCell: Cell;lHandle: ListHandle): BOOLEAN;
```

```
009058      INLINE $3F3C,$0048,$A9E7;
009059  FUNCTION LSearch(dataPtr: Ptr;dataLen: INTEGER;searchProc: ProcPtr;VAR theCell: Cell;
009060      lHandle: ListHandle): BOOLEAN;
009061      INLINE $3F3C,$0054,$A9E7;
009062  PROCEDURE LSize(listWidth: INTEGER;listHeight: INTEGER;lHandle: ListHandle);
009063      INLINE $3F3C,$0060,$A9E7;
009064  PROCEDURE LDoDraw(drawIt: BOOLEAN;lHandle: ListHandle);
009065      INLINE $3F3C,$002C,$A9E7;
009066  PROCEDURE LScroll(dCols: INTEGER;dRows: INTEGER;lHandle: ListHandle);
009067      INLINE $3F3C,$0050,$A9E7;
009068  PROCEDURE LAutoScroll(lHandle: ListHandle);
009069      INLINE $3F3C,$0010,$A9E7;
009070  PROCEDURE LUpdate(theRgn: RgnHandle;lHandle: ListHandle);
009071      INLINE $3F3C,$0064,$A9E7;
009072  PROCEDURE LActivate(act: BOOLEAN;lHandle: ListHandle);
009073      INLINE $4267,$A9E7;
009074  PROCEDURE LCellSize(cSize: Point;lHandle: ListHandle);
009075      INLINE $3F3C,$0014,$A9E7;
009076  FUNCTION LClick(pt: Point;modifiers: INTEGER;lHandle: ListHandle): BOOLEAN;
009077      INLINE $3F3C,$0018,$A9E7;
009078  PROCEDURE LAddToCell(dataPtr: Ptr;dataLen: INTEGER;theCell: Cell;lHandle: ListHandle);
009079      INLINE $3F3C,$000C,$A9E7;
009080  PROCEDURE LClrCell(theCell: Cell;lHandle: ListHandle);
009081      INLINE $3F3C,$001C,$A9E7;
009082  PROCEDURE LGetCell(dataPtr: Ptr;VAR dataLen: INTEGER;theCell: Cell;lHandle: ListHandle);
009083      INLINE $3F3C,$0038,$A9E7;
009084  PROCEDURE LFind(VAR offset: INTEGER;VAR len: INTEGER;theCell: Cell;lHandle: ListHandle);
009085      INLINE $3F3C,$0034,$A9E7;
009086  PROCEDURE LRect(VAR cellRect: Rect;theCell: Cell;lHandle: ListHandle);
009087      INLINE $3F3C,$004C,$A9E7;
009088  PROCEDURE LSetCell(dataPtr: Ptr;dataLen: INTEGER;theCell: Cell;lHandle: ListHandle);
009089      INLINE $3F3C,$0058,$A9E7;
009090  PROCEDURE LSetSelect(setIt: BOOLEAN;theCell: Cell;lHandle: ListHandle);
009091      INLINE $3F3C,$005C,$A9E7;
009092  PROCEDURE LDraw(theCell: Cell;lHandle: ListHandle);
009093      INLINE $3F3C,$0030,$A9E7;
009094
009095
009096  {$ENDC}    { UsingLists }
009097
009098  {$IFC NOT UsingIncludes}
009099      END.
009100  {$ENDC}
009101
009102
009103  ### END OF FILE Lists.p
009104
```

```
009105
009106  ############################################################################
009107  ### FILE: MacPrint.p
009108  ############################################################################
009109
009110  {
009111      File: MacPrint.p
009112
009113      As of MPW 3.0, interface files were reorganized to more closely
009114      match "Inside Macintosh" reference books and be more consistant
009115      from language to language.
009116
009117      Interfaces for the non-ROM based Print Manager are now found in Printing.p.
009118      This file, which includes Printing.p, is provided for compatibility
009119      with old sources.
009120
009121      Pascal Interface to the Macintosh Libraries
009122      Copyright Apple Computer, Inc. 1988
009123      All Rights Reserved
009124  }
009125
009126  {$IFC UNDEFINED UsingIncludes}
009127  {$SETC UsingIncludes := 0}
009128  {$ENDC}
009129
009130  {$IFC NOT UsingIncludes}
009131      UNIT MacPrint;
009132      INTERFACE
009133  {$ENDC}
009134
009135  {$IFC UNDEFINED UsingMacPrint}
009136  {$SETC UsingMacPrint := 1}
009137
009138  {$I+}
009139  {$SETC MacPrintIncludes := UsingIncludes}
009140  {$SETC UsingIncludes := 1}
009141  {$IFC UNDEFINED UsingPrinting}
009142  {$I $$Shell(PInterfaces)Printing.p}
009143  {$ENDC}
009144  {$SETC UsingIncludes := MacPrintIncludes}
009145
009146  {$ENDC}    { UsingMacPrint }
009147
009148  {$IFC NOT UsingIncludes}
009149      END.
009150  {$ENDC}
009151
009152
009153  ### END OF FILE MacPrint.p
009154
```

```
009155
009156  ############################################################################
009157  ### FILE: Memory.p
009158  ############################################################################
009159
009160  {
009161  Created: Sunday, January 6, 1991 at 10:47 PM
009162      Memory.p
009163      Pascal Interface to the Macintosh Libraries
009164
009165          Copyright Apple Computer, Inc.    1985-1990
009166          All rights reserved
009167  }
009168
009169
009170  {$IFC UNDEFINED UsingIncludes}
009171  {$SETC UsingIncludes := 0}
009172  {$ENDC}
009173
009174  {$IFC NOT UsingIncludes}
009175      UNIT Memory;
009176      INTERFACE
009177  {$ENDC}
009178
009179  {$IFC UNDEFINED UsingMemory}
009180  {$SETC UsingMemory := 1}
009181
009182  {$I+}
009183  {$SETC MemoryIncludes := UsingIncludes}
009184  {$SETC UsingIncludes := 1}
009185  {$IFC UNDEFINED UsingTypes}
009186  {$I $$Shell(PInterfaces)Types.p}
009187  {$ENDC}
009188  {$SETC UsingIncludes := MemoryIncludes}
009189
009190  CONST
009191  maxSize = $800000;  {Max data block size is 8 megabytes}
009192  defaultPhysicalEntryCount = 8;
009193
009194  { values returned from the GetPageState function }
009195  kPageInMemory = 0;
009196  kPageOnDisk = 1;
009197  kNotPaged = 2;
009198
009199  TYPE
009200  Size = LONGINT;     { size of a block in bytes }
009201
009202  THz = ^Zone;
009203  Zone = RECORD
009204      bkLim: Ptr;
009205      purgePtr: Ptr;
009206      hFstFree: Ptr;
009207      zcbFree: LONGINT;
009208      gzProc: ProcPtr;
009209      moreMast: INTEGER;
009210      flags: INTEGER;
```

```
009211      cntRel: INTEGER;
009212      maxRel: INTEGER;
009213      cntNRel: INTEGER;
009214      maxNRel: INTEGER;
009215      cntEmpty: INTEGER;
009216      cntHandles: INTEGER;
009217      minCBFree: LONGINT;
009218      purgeProc: ProcPtr;
009219      sparePtr: Ptr;
009220      allocPtr: Ptr;
009221      heapData: INTEGER;
009222      END;
009223
009224  MemoryBlock = RECORD
009225      address: Ptr;
009226      count: LONGINT;
009227      END;
009228
009229  LogicalToPhysicalTable = RECORD
009230      logical: MemoryBlock;
009231      physical: ARRAY [0..defaultPhysicalEntryCount - 1] OF MemoryBlock;
009232      END;
009233
009234
009235  PageState = INTEGER;
009236  StatusRegisterContents = INTEGER;
009237
009238  FUNCTION GetApplLimit: Ptr;
009239      INLINE $2EB8,$0130;
009240  FUNCTION GetZone: THz;
009241      INLINE $A11A,$2E88;
009242  FUNCTION SystemZone: THz;
009243      INLINE $2EB8,$02A6;
009244  FUNCTION ApplicZone: THz;
009245      INLINE $2EB8,$02AA;
009246  FUNCTION ApplicationZone: THz;
009247      INLINE $2EB8,$02AA;
009248  FUNCTION NewHandle(byteCount: Size): Handle;
009249  FUNCTION NewHandleSys(byteCount: Size): Handle;
009250  FUNCTION NewHandleClear(byteCount: Size): Handle;
009251  FUNCTION NewHandleSysClear(byteCount: Size): Handle;
009252  FUNCTION HandleZone(h: Handle): THz;
009253  FUNCTION RecoverHandle(p: Ptr): Handle;
009254  FUNCTION NewPtr(byteCount: Size): Ptr;
009255  FUNCTION NewPtrSys(byteCount: Size): Ptr;
009256  FUNCTION NewPtrClear(byteCount: Size): Ptr;
009257  FUNCTION NewPtrSysClear(byteCount: Size): Ptr;
009258  FUNCTION PtrZone(p: Ptr): THz;
009259  FUNCTION GZSaveHnd: Handle;
009260      INLINE $2EB8,$0328;
009261  FUNCTION TopMem: Ptr;
009262      INLINE $2EB8,$0108;
009263  FUNCTION MaxBlock: LONGINT;
009264  FUNCTION MaxBlockSys: LONGINT;
009265  FUNCTION StackSpace: LONGINT;
009266  FUNCTION NewEmptyHandle: Handle;
```

```
009267  FUNCTION NewEmptyHandleSys: Handle;
009268  PROCEDURE HLock(h: Handle);
009269      INLINE $205F,$A029;
009270  PROCEDURE HUnlock(h: Handle);
009271      INLINE $205F,$A02A;
009272  PROCEDURE HPurge(h: Handle);
009273      INLINE $205F,$A049;
009274  PROCEDURE HNoPurge(h: Handle);
009275      INLINE $205F,$A04A;
009276  PROCEDURE HLockHi(h: Handle);
009277      INLINE $205F,$A064,$A029;
009278  FUNCTION StripAddress(theAddress: UNIV Ptr): Ptr;
009279  {$IFC SystemSixOrLater }
009280      INLINE $201F,$A055,$2E80;
009281  {$ENDC}
009282  FUNCTION Translate24To32(addr24: UNIV Ptr): Ptr;
009283      INLINE $201F,$A091,$2E80;
009284  FUNCTION TempNewHandle(logicalSize: Size;VAR resultCode: OSErr): Handle;
009285      INLINE $3F3C,$001D,$A88F;
009286  FUNCTION TempMaxMem(VAR grow: Size): Size;
009287      INLINE $3F3C,$0015,$A88F;
009288  FUNCTION TempFreeMem: LONGINT;
009289      INLINE $3F3C,$0018,$A88F;
009290
009291  {  Temporary Memory routines renamed, but obsolete, in System 7.0 and later.  }
009292  PROCEDURE TempHLock(h: Handle;VAR resultCode: OSErr);
009293      INLINE $3F3C,$001E,$A88F;
009294  PROCEDURE TempHUnlock(h: Handle;VAR resultCode: OSErr);
009295      INLINE $3F3C,$001F,$A88F;
009296  PROCEDURE TempDisposeHandle(h: Handle;VAR resultCode: OSErr);
009297      INLINE $3F3C,$0020,$A88F;
009298  FUNCTION TempTopMem: Ptr;
009299      INLINE $3F3C,$0016,$A88F;
009300
009301  {  Temporary Memory routines as they were known before System 7.0.  }
009302  FUNCTION MFMaxMem(VAR grow: Size): Size;
009303      INLINE $3F3C,$0015,$A88F;
009304  FUNCTION MFFreeMem: LONGINT;
009305      INLINE $3F3C,$0018,$A88F;
009306  FUNCTION MFTempNewHandle(logicalSize: Size;VAR resultCode: OSErr): Handle;
009307      INLINE $3F3C,$001D,$A88F;
009308  PROCEDURE MFTempHLock(h: Handle;VAR resultCode: OSErr);
009309      INLINE $3F3C,$001E,$A88F;
009310  PROCEDURE MFTempHUnlock(h: Handle;VAR resultCode: OSErr);
009311      INLINE $3F3C,$001F,$A88F;
009312  PROCEDURE MFTempDisposHandle(h: Handle;VAR resultCode: OSErr);
009313      INLINE $3F3C,$0020,$A88F;
009314  FUNCTION MFTopMem: Ptr;
009315      INLINE $3F3C,$0016,$A88F;
009316  PROCEDURE InitApplZone;
009317      INLINE $A02C;
009318  PROCEDURE InitZone(pgrowZone: ProcPtr;cmoreMasters: INTEGER;limitPtr: UNIV Ptr;
009319      startPtr: UNIV Ptr);
009320  PROCEDURE SetZone(hz: THz);
009321      INLINE $205F,$A01B;
009322  FUNCTION CompactMem(cbNeeded: Size): Size;
```

```
009323  FUNCTION CompactMemSys(cbNeeded: Size): Size;
009324  PROCEDURE PurgeMem(cbNeeded: Size);
009325      INLINE $201F,$A04D;
009326  PROCEDURE PurgeMemSys(cbNeeded: Size);
009327      INLINE $201F,$A44D;
009328  FUNCTION FreeMem: LONGINT;
009329      INLINE $A01C,$2E80;
009330  FUNCTION FreeMemSys: LONGINT;
009331      INLINE $A41C,$2E80;
009332  PROCEDURE ResrvMem(cbNeeded: Size);
009333      INLINE $201F,$A040;
009334  PROCEDURE ReserveMem(cbNeeded: Size);
009335      INLINE $201F,$A040;
009336  PROCEDURE ReserveMemSys(cbNeeded: Size);
009337      INLINE $201F,$A440;
009338  FUNCTION MaxMem(VAR grow: Size): Size;
009339  FUNCTION MaxMemSys(VAR grow: Size): Size;
009340  PROCEDURE SetGrowZone(growZone: ProcPtr);
009341      INLINE $205F,$A04B;
009342  PROCEDURE SetApplLimit(zoneLimit: UNIV Ptr);
009343      INLINE $205F,$A02D;
009344  PROCEDURE MoveHHi(h: Handle);
009345      INLINE $205F,$A064;
009346  PROCEDURE DisposPtr(p: Ptr);
009347      INLINE $205F,$A01F;
009348  PROCEDURE DisposePtr(p: Ptr);
009349      INLINE $205F,$A01F;
009350  FUNCTION GetPtrSize(p: Ptr): Size;
009351  PROCEDURE SetPtrSize(p: Ptr;newSize: Size);
009352  PROCEDURE DisposHandle(h: Handle);
009353      INLINE $205F,$A023;
009354  PROCEDURE DisposeHandle(h: Handle);
009355      INLINE $205F,$A023;
009356  FUNCTION GetHandleSize(h: Handle): Size;
009357  PROCEDURE SetHandleSize(h: Handle;newSize: Size);
009358  PROCEDURE EmptyHandle(h: Handle);
009359      INLINE $205F,$A02B;
009360  PROCEDURE ReallocHandle(h: Handle;byteCount: Size);
009361  PROCEDURE ReallocateHandle(h: Handle;byteCount: Size);
009362  PROCEDURE HSetRBit(h: Handle);
009363      INLINE $205F,$A067;
009364  PROCEDURE HClrRBit(h: Handle);
009365      INLINE $205F,$A068;
009366  PROCEDURE MoreMasters;
009367      INLINE $A036;
009368  PROCEDURE BlockMove(srcPtr: UNIV Ptr;destPtr: UNIV Ptr;byteCount: Size);
009369  FUNCTION MemError: OSErr;
009370      INLINE $3EB8,$0220;
009371  PROCEDURE PurgeSpace(VAR total: LONGINT;VAR contig: LONGINT);
009372  FUNCTION HGetState(h: Handle): SignedByte;
009373  PROCEDURE HSetState(h: Handle;flags: SignedByte);
009374  PROCEDURE SetApplBase(startPtr: UNIV Ptr);
009375      INLINE $205F,$A057;
009376  PROCEDURE MaxApplZone;
009377      INLINE $A063;
009378  FUNCTION HoldMemory(address: UNIV Ptr;count: LONGINT): OSErr;
```

```
009379  FUNCTION UnholdMemory(address: UNIV Ptr;count: LONGINT): OSErr;
009380  FUNCTION LockMemory(address: UNIV Ptr;count: LONGINT): OSErr;
009381  FUNCTION LockMemoryContiguous(address: UNIV Ptr;count: LONGINT): OSErr;
009382  FUNCTION UnlockMemory(address: UNIV Ptr;count: LONGINT): OSErr;
009383  FUNCTION GetPhysical(VAR addresses: LogicalToPhysicalTable;VAR physicalEntryCount: LONGINT): OSErr;
009384  FUNCTION DeferUserFn(userFunction: ProcPtr;argument: UNIV Ptr): OSErr;
009385  FUNCTION DebuggerGetMax: LONGINT;
009386      INLINE $7000,$A08D,$2E80;
009387  PROCEDURE DebuggerEnter;
009388      INLINE $7001,$A08D;
009389  PROCEDURE DebuggerExit;
009390      INLINE $7002,$A08D;
009391  PROCEDURE DebuggerPoll;
009392      INLINE $7003,$A08D;
009393  FUNCTION GetPageState(address: UNIV Ptr): PageState;
009394      INLINE $205F,$7004,$A08D,$3E80;
009395  FUNCTION PageFaultFatal: BOOLEAN;
009396      INLINE $7005,$A08D,$1E80;
009397  FUNCTION DebuggerLockMemory(address: UNIV Ptr;count: LONGINT): OSErr;
009398  FUNCTION DebuggerUnlockMemory(address: UNIV Ptr;count: LONGINT): OSErr;
009399  FUNCTION EnterSupervisorMode: StatusRegisterContents;
009400      INLINE $7008,$A08D,$3E80;
009401
009402
009403  {$ENDC}    { UsingMemory }
009404
009405  {$IFC NOT UsingIncludes}
009406      END.
009407  {$ENDC}
009408
009409
009410  ### END OF FILE Memory.p
009411
```

```
009412
009413  ************************************************************************
009414  *** FILE: MemTypes.p
009415  ************************************************************************
009416
009417  {
009418      File: MemTypes.p
009419
009420      As of MPW 3.0, interface files were reorganized to more closely
009421      match "Inside Macintosh" reference books and be more consistant
009422      from language to language.
009423
009424      Interfaces for the basic type definitions are now found in Types.p.
009425      This file, which includes Types.p, is provided for compatibility
009426      with old sources.
009427
009428      Pascal Interface to the Macintosh Libraries
009429      Copyright Apple Computer, Inc. 1988
009430      All Rights Reserved
009431  }
009432
009433  {$IFC UNDEFINED UsingIncludes}
009434  {$SETC UsingIncludes := 0}
009435  {$ENDC}
009436
009437  {$IFC NOT UsingIncludes}
009438      UNIT MemTypes;
009439      INTERFACE
009440  {$ENDC}
009441
009442  {$IFC UNDEFINED UsingMemTypes}
009443  {$SETC UsingMemTypes := 1}
009444
009445  {$I+}
009446  {$SETC MemTypesIncludes := UsingIncludes}
009447  {$SETC UsingIncludes := 1}
009448  {$IFC UNDEFINED UsingTypes}
009449  {$I $$Shell(PInterfaces)Types.p}
009450  {$ENDC}
009451  {$SETC UsingIncludes := MemTypesIncludes}
009452
009453  {$ENDC}    { UsingMemTypes }
009454
009455  {$IFC NOT UsingIncludes}
009456      END.
009457  {$ENDC}
009458
009459
009460  ### END OF FILE MemTypes.p
009461
```

```
009462
009463  ##########################################################################
009464  ### FILE: Menus.p
009465  ##########################################################################
009466
009467  {
009468  Created: Sunday, January 6, 1991 at 10:48 PM
009469      Menus.p
009470      Pascal Interface to the Macintosh Libraries
009471
009472          Copyright Apple Computer, Inc.    1985-1990
009473          All rights reserved
009474  }
009475
009476
009477  {$IFC UNDEFINED UsingIncludes}
009478  {$SETC UsingIncludes := 0}
009479  {$ENDC}
009480
009481  {$IFC NOT UsingIncludes}
009482      UNIT Menus;
009483      INTERFACE
009484  {$ENDC}
009485
009486  {$IFC UNDEFINED UsingMenus}
009487  {$SETC UsingMenus := 1}
009488
009489  {$I+}
009490  {$SETC MenusIncludes := UsingIncludes}
009491  {$SETC UsingIncludes := 1}
009492  {$IFC UNDEFINED UsingQuickdraw}
009493  {$I $$Shell(PInterfaces)Quickdraw.p}
009494  {$ENDC}
009495  {$SETC UsingIncludes := MenusIncludes}
009496
009497  CONST
009498  noMark = 0;                             {mark symbol for MarkItem}
009499
009500  { menu defProc messages }
009501  mDrawMsg = 0;
009502  mChooseMsg = 1;
009503  mSizeMsg = 2;
009504  mDrawItemMsg = 4;
009505  mCalcItemMsg = 5;
009506  textMenuProc = 0;
009507  hMenuCmd = 27;                          {itemCmd == 0x001B ==> hierarchical menu}
009508  hierMenu = -1;                          {a hierarchical menu - for InsertMenu call}
009509  mPopUpMsg = 3;                          {menu defProc messages - place yourself}
009510  mctAllItems = -98;                      {search for all Items for the given ID}
009511  mctLastIDIndic = -99;                   {last color table entry has this in ID field}
009512
009513  TYPE
009514  MenuPtr = ^MenuInfo;
009515  MenuHandle = ^MenuPtr;
009516  MenuInfo = RECORD
009517      menuID: INTEGER;
```

```
009518      menuWidth: INTEGER;
009519      menuHeight: INTEGER;
009520      menuProc: Handle;
009521      enableFlags: LONGINT;
009522      menuData: Str255;
009523      END;
009524
009525  MCEntryPtr = ^MCEntry;
009526  MCEntry = RECORD
009527      mctID: INTEGER;                     {menu ID.  ID = 0 is the menu bar}
009528      mctItem: INTEGER;                   {menu Item. Item = 0 is a title}
009529      mctRGB1: RGBColor;                  {usage depends on ID and Item}
009530      mctRGB2: RGBColor;                  {usage depends on ID and Item}
009531      mctRGB3: RGBColor;                  {usage depends on ID and Item}
009532      mctRGB4: RGBColor;                  {usage depends on ID and Item}
009533      mctReserved: INTEGER;               {reserved for internal use}
009534      END;
009535
009536
009537  {}
009538
009539  MCTablePtr = ^MCTable;
009540  MCTableHandle = ^MCTablePtr;
009541
009542  MCTable = ARRAY [0..0] OF MCEntry;      { the entries themselves }
009543
009544  MenuCRsrcPtr = ^MenuCRsrc;
009545  MenuCRsrcHandle = ^MenuCRsrcPtr;
009546  MenuCRsrc = RECORD
009547      numEntries: INTEGER;                {number of entries}
009548      mcEntryRecs: MCTable;               {ARRAY [1..numEntries] of MCEntry}
009549      END;
009550
009551
009552  PROCEDURE InitMenus;
009553      INLINE $A930;
009554  FUNCTION NewMenu(menuID: INTEGER;menuTitle: Str255): MenuHandle;
009555      INLINE $A931;
009556  FUNCTION GetMenu(resourceID: INTEGER): MenuHandle;
009557      INLINE $A9BF;
009558  PROCEDURE DisposeMenu(theMenu: MenuHandle);
009559      INLINE $A932;
009560  PROCEDURE AppendMenu(menu: MenuHandle;data: Str255);
009561      INLINE $A933;
009562  PROCEDURE AddResMenu(theMenu: MenuHandle;theType: ResType);
009563      INLINE $A94D;
009564  PROCEDURE InsertResMenu(theMenu: MenuHandle;theType: ResType;afterItem: INTEGER);
009565      INLINE $A951;
009566  PROCEDURE InsertMenu(theMenu: MenuHandle;beforeID: INTEGER);
009567      INLINE $A935;
009568  PROCEDURE DrawMenuBar;
009569      INLINE $A937;
009570  PROCEDURE InvalMenuBar;
009571      INLINE $A81D;
009572  PROCEDURE DeleteMenu(menuID: INTEGER);
009573      INLINE $A936;
```

```
009574  PROCEDURE ClearMenuBar;
009575      INLINE $A934;
009576  FUNCTION GetNewMBar(menuBarID: INTEGER): Handle;
009577      INLINE $A9C0;
009578  FUNCTION GetMenuBar: Handle;
009579      INLINE $A93B;
009580  PROCEDURE SetMenuBar(menuList: Handle);
009581      INLINE $A93C;
009582  PROCEDURE InsMenuItem(theMenu: MenuHandle;itemString: Str255;afterItem: INTEGER);
009583      INLINE $A826;
009584  PROCEDURE DelMenuItem(theMenu: MenuHandle;item: INTEGER);
009585      INLINE $A952;
009586  FUNCTION MenuKey(ch: CHAR): LONGINT;
009587      INLINE $A93E;
009588  PROCEDURE HiliteMenu(menuID: INTEGER);
009589      INLINE $A938;
009590  PROCEDURE SetItem(theMenu: MenuHandle;item: INTEGER;itemString: Str255);
009591      INLINE $A947;
009592  PROCEDURE GetItem(theMenu: MenuHandle;item: INTEGER;VAR itemString: Str255);
009593      INLINE $A946;
009594  PROCEDURE DisableItem(theMenu: MenuHandle;item: INTEGER);
009595      INLINE $A93A;
009596  PROCEDURE EnableItem(theMenu: MenuHandle;item: INTEGER);
009597      INLINE $A939;
009598  PROCEDURE CheckItem(theMenu: MenuHandle;item: INTEGER;checked: BOOLEAN);
009599      INLINE $A945;
009600  PROCEDURE SetItemMark(theMenu: MenuHandle;item: INTEGER;markChar: CHAR);
009601      INLINE $A944;
009602  PROCEDURE GetItemMark(theMenu: MenuHandle;item: INTEGER;VAR markChar: CHAR);
009603      INLINE $A943;
009604  PROCEDURE SetItemIcon(theMenu: MenuHandle;item: INTEGER;iconIndex: Byte);
009605      INLINE $A940;
009606  PROCEDURE GetItemIcon(theMenu: MenuHandle;item: INTEGER;VAR iconIndex: Byte);
009607      INLINE $A93F;
009608  PROCEDURE SetItemStyle(theMenu: MenuHandle;item: INTEGER;chStyle: Style);
009609      INLINE $A942;
009610  PROCEDURE GetItemStyle(theMenu: MenuHandle;item: INTEGER;VAR chStyle: Style);
009611  PROCEDURE CalcMenuSize(theMenu: MenuHandle);
009612      INLINE $A948;
009613  FUNCTION CountMItems(theMenu: MenuHandle): INTEGER;
009614      INLINE $A950;
009615  FUNCTION GetMHandle(menuID: INTEGER): MenuHandle;
009616      INLINE $A949;
009617  PROCEDURE FlashMenuBar(menuID: INTEGER);
009618      INLINE $A94C;
009619  PROCEDURE SetMenuFlash(count: INTEGER);
009620      INLINE $A94A;
009621  FUNCTION MenuSelect(startPt: Point): LONGINT;
009622      INLINE $A93D;
009623  PROCEDURE InitProcMenu(resID: INTEGER);
009624      INLINE $A808;
009625  PROCEDURE GetItemCmd(theMenu: MenuHandle;item: INTEGER;VAR cmdChar: CHAR);
009626      INLINE $A84E;
009627  PROCEDURE SetItemCmd(theMenu: MenuHandle;item: INTEGER;cmdChar: CHAR);
009628      INLINE $A84F;
009629  FUNCTION PopUpMenuSelect(menu: MenuHandle;top: INTEGER;left: INTEGER;popUpItem: INTEGER): LONGINT;
```

```
009630      INLINE $A80B;
009631  FUNCTION MenuChoice: LONGINT;
009632      INLINE $AA66;
009633  PROCEDURE DelMCEntries(menuID: INTEGER;menuItem: INTEGER);
009634      INLINE $AA60;
009635  FUNCTION GetMCInfo: MCTableHandle;
009636      INLINE $AA61;
009637  PROCEDURE SetMCInfo(menuCTbl: MCTableHandle);
009638      INLINE $AA62;
009639  PROCEDURE DispMCInfo(menuCTbl: MCTableHandle);
009640      INLINE $AA63;
009641  FUNCTION GetMCEntry(menuID: INTEGER;menuItem: INTEGER): MCEntryPtr;
009642      INLINE $AA64;
009643  PROCEDURE SetMCEntries(numEntries: INTEGER;menuCEntries: MCTablePtr);
009644      INLINE $AA65;
009645
009646
009647  {$ENDC}    { UsingMenus }
009648
009649  {$IFC NOT UsingIncludes}
009650      END.
009651  {$ENDC}
009652
009653
009654  ### END OF FILE Menus.p
009655
```

```
009656
009657  ############################################################################
009658  ### FILE: MIDI.p
009659  ############################################################################
009660
009661
009662  {
009663  Created: Tuesday, November 26, 1991 at 3:28 PM
009664   MIDI.p
009665   Pascal Interface to the Macintosh Libraries
009666
009667    Copyright Apple Computer, Inc. 1988-1991
009668    All rights reserved
009669  }
009670
009671
009672  {$IFC UNDEFINED UsingIncludes}
009673  {$SETC UsingIncludes := 0}
009674  {$ENDC}
009675
009676  {$IFC NOT UsingIncludes}
009677   UNIT MIDI;
009678   INTERFACE
009679  {$ENDC}
009680
009681  {$IFC UNDEFINED UsingMIDI}
009682  {$SETC UsingMIDI := 1}
009683
009684  {$I+}
009685  {$SETC MIDIIncludes := UsingIncludes}
009686  {$SETC UsingIncludes := 1}
009687  {$IFC UNDEFINED UsingTypes}
009688  {$I $$Shell(PInterfaces)Types.p}
009689  {$ENDC}
009690  {$SETC UsingIncludes := MIDIIncludes}
009691
009692  CONST
009693  midiToolNum = 4;                    {tool number of MIDI Manager for SndDispVersion call}
009694  midiMaxNameLen = 31;                {maximum number of characters in port and client names}
009695
009696  { Time formats }
009697  midiFormatMSec = 0;                 {milliseconds}
009698  midiFormatBeats = 1;                {beats}
009699  midiFormat24fpsBit = 2;             {24 frames/sec.}
009700  midiFormat25fpsBit = 3;             {25 frames/sec.}
009701  midiFormat30fpsDBit = 4;            {30 frames/sec. drop-frame}
009702  midiFormat30fpsBit = 5;             {30 frames/sec.}
009703  midiFormat24fpsQF = 6;              {24 frames/sec. longInt format }
009704  midiFormat25fpsQF = 7;              {25 frames/sec. longInt format }
009705  midiFormat30fpsDQF = 8;             {30 frames/sec. drop-frame longInt format }
009706  midiFormat30fpsQF = 9;              {30 frames/sec. longInt format }
009707  midiInternalSync = 0;               {internal sync}
009708  midiExternalSync = 1;               {external sync}
009709
009710  { Port types}
009711  midiPortTypeTime = 0;               {time port}
```

```
009712  midiPortTypeInput = 1;              {input port}
009713  midiPortTypeOutput = 2;             {output port}
009714  midiPortTypeTimeInv = 3;            {invisible time port}
009715
009716  { OffsetTimes  }
009717  midiGetEverything = $7FFFFFFF;      {get all packets, regardless of time stamps}
009718  midiGetNothing = $80000000;         {get no packets, regardless of time stamps}
009719  midiGetCurrent = $00000000;         {get current packets only}
009720
009721  {    MIDI data and messages are passed in MIDIPacket records (see below).
009722      The first byte of every MIDIPacket contains a set of flags
009723
009724      bits 0-1    00 = new MIDIPacket, not continued
009725                       01 = begining of continued MIDIPacket
009726                       10 = end of continued MIDIPacket
009727                      11 = continuation
009728      bits 2-3     reserved
009729
009730      bits 4-6      000 = packet contains MIDI data
009731
009732                        001 = packet contains MIDI Manager message
009733
009734      bit 7               0 = MIDIPacket has valid stamp
009735                           1 = stamp with current clock }
009736  midiContMask = $03;
009737  midiNoCont = $00;
009738  midiStartCont = $01;
009739  midiMidCont = $03;
009740  midiEndCont = $02;
009741  midiTypeMask = $70;
009742  midiMsgType = $00;
009743  midiMgrType = $10;
009744  midiTimeStampMask = $80;
009745  midiTimeStampCurrent = $80;
009746  midiTimeStampValid = $00;
009747
009748  {    MIDI Manager MIDIPacket command words (the first word in the data field
009749      for midiMgrType messages) }
009750  midiOverflowErr = $0001;
009751  midiSCCErr = $0002;
009752  midiPacketErr = $0003;
009753  midiMaxErr = $00FF;                 {all command words less than this value  are error indicators}
009754
009755  { Valid results to be returned by readHooks }
009756  midiKeepPacket = 0;
009757  midiMorePacket = 1;
009758  midiNoMorePacket = 2;
009759
009760  { Errors: }
009761  midiNoClientErr = -250;             {no client with that ID found}
009762  midiNoPortErr = -251;               {no port with that ID found}
009763  midiTooManyPortsErr = -252;         {too many ports already installed in the system}
009764  midiTooManyConsErr = -253;          {too many connections made}
009765  midiVConnectErr = -254;             {pending virtual connection created}
009766  midiVConnectMade = -255;            {pending virtual connection resolved}
009767  midiVConnectRmvd = -256;            {pending virtual connection removed}
```

```
009768  midiNoConErr = -257;               {no connection exists between specified ports}
009769  midiWriteErr = -258;               {MIDIWritePacket couldn't write to all connected ports}
009770  midiNameLenErr = -259;             {name supplied is longer than 31 characters}
009771  midiDupIDErr = -260;               {duplicate client ID}
009772  midiInvalidCmdErr = -261;          {command not supported for port type}
009773
009774  {     Driver calls: }
009775  midiOpenDriver = 1;
009776  midiCloseDriver = 2;
009777
009778  TYPE
009779  MIDIPacketPtr = ^MIDIPacket;
009780  MIDIPacket = PACKED RECORD
009781   flags: Byte;
009782   len: Byte;
009783   tStamp: LONGINT;
009784   data: PACKED ARRAY [0..248] OF Byte;
009785   END;
009786
009787  MIDIClkInfo = RECORD
009788   sync: INTEGER;                    {synchronization external/internal}
009789   curTime: LONGINT;                 {current value of port's clock}
009790   format: INTEGER;                  {time code format}
009791   END;
009792
009793  MIDIIDRec = RECORD
009794   clientID: OSType;
009795   portID: OSType;
009796   END;
009797
009798  MIDIPortInfoPtr = ^MIDIPortInfo;
009799  MIDIPortInfoHdl = ^MIDIPortInfoPtr;
009800  MIDIPortInfo = RECORD
009801   portType: INTEGER;                {type of port}
009802   timeBase: MIDIIDRec;              {MIDIIDRec for time base}
009803   numConnects: INTEGER;             {number of connections}
009804   cList: ARRAY [1..1] OF MIDIIDRec;  {-r or $R- permits access to [1..numConnects] of MIDIIDRec}
009805   END;
009806
009807  MIDIPortParamsPtr = ^MIDIPortParams;
009808  MIDIPortParams = RECORD
009809   portID: OSType;                   {ID of port, unique within client}
009810   portType: INTEGER;                {Type of port - input, output, time, etc.}
009811   timeBase: INTEGER;                {refnum of time base, 0 if none}
009812   offsetTime: LONGINT;              {offset for current time stamps}
009813   readHook: Ptr;                    {routine to call when input data is valid}
009814   refCon: LONGINT;                  {refcon for port (for client use)}
009815   initClock: MIDIClkInfo;           {initial settings for a time base}
009816   name: Str255;                     {name of the port, This is a real live string, not a ptr.}
009817   END;
009818
009819  MIDIIDListPtr = ^MIDIIDList;
009820  MIDIIDListHdl = ^MIDIIDListPtr;
009821  MIDIIDList = RECORD
009822   numIDs: INTEGER;
009823   list: ARRAY [1..1] OF OSType;      { -r or $R- permits access to [1..numIDs] of OSType }
```

```
009824   END;
009825
009826
009827  {
009828       Prototype Declarations for readHook and timeProc
009829
009830       FUNCTION myReadHook(myPacket: MIDIPacketPtr; myRefCon: LONGINT) : INTEGER;
009831       PROCEDURE myTimeProc(curTime: LONGINT; myRefCon: LONGINT);
009832  }
009833
009834  FUNCTION SndDispVersion(toolnum: INTEGER): LONGINT;
009835  FUNCTION MIDISignIn(clientID: OSType;refCon: LONGINT;icon: Handle;name: Str255): OSErr;
009836   INLINE $203C,$0004,midiToolNum,$A800;
009837  PROCEDURE MIDISignOut(clientID: OSType);
009838   INLINE $203C,$0008,midiToolNum,$A800;
009839  FUNCTION MIDIGetClients: MIDIIDListHdl;
009840   INLINE $203C,$000C,midiToolNum,$A800;
009841  PROCEDURE MIDIGetClientName(clientID: OSType;VAR name: Str255);
009842   INLINE $203C,$0010,midiToolNum,$A800;
009843  PROCEDURE MIDISetClientName(clientID: OSType;name: Str255);
009844   INLINE $203C,$0014,midiToolNum,$A800;
009845  FUNCTION MIDIGetPorts(clientID: OSType): MIDIIDListHdl;
009846   INLINE $203C,$0018,midiToolNum,$A800;
009847  FUNCTION MIDIAddPort(clientID: OSType;BufSize: INTEGER;VAR refnum: INTEGER;
009848   init: MIDIPortParamsPtr): OSErr;
009849   INLINE $203C,$001C,midiToolNum,$A800;
009850  FUNCTION MIDIGetPortInfo(clientID: OSType;portID: OSType): MIDIPortInfoHdl;
009851   INLINE $203C,$0020,midiToolNum,$A800;
009852  FUNCTION MIDIConnectData(srcClID: OSType;srcPortID: OSType;dstClID: OSType;
009853   dstPortID: OSType): OSErr;
009854   INLINE $203C,$0024,midiToolNum,$A800;
009855  FUNCTION MIDIUnConnectData(srcClID: OSType;srcPortID: OSType;dstClID: OSType;
009856   dstPortID: OSType): OSErr;
009857   INLINE $203C,$0028,midiToolNum,$A800;
009858  FUNCTION MIDIConnectTime(srcClID: OSType;srcPortID: OSType;dstClID: OSType;
009859   dstPortID: OSType): OSErr;
009860   INLINE $203C,$002C,midiToolNum,$A800;
009861  FUNCTION MIDIUnConnectTime(srcClID: OSType;srcPortID: OSType;dstClID: OSType;
009862   dstPortID: OSType): OSErr;
009863   INLINE $203C,$0030,midiToolNum,$A800;
009864  PROCEDURE MIDIFlush(refnum: INTEGER);
009865   INLINE $203C,$0034,midiToolNum,$A800;
009866  FUNCTION MIDIGetReadHook(refnum: INTEGER): ProcPtr;
009867   INLINE $203C,$0038,midiToolNum,$A800;
009868  PROCEDURE MIDISetReadHook(refnum: INTEGER;hook: ProcPtr);
009869   INLINE $203C,$003C,midiToolNum,$A800;
009870  PROCEDURE MIDIGetPortName(clientID: OSType;portID: OSType;VAR name: Str255);
009871   INLINE $203C,$0040,midiToolNum,$A800;
009872  PROCEDURE MIDISetPortName(clientID: OSType;portID: OSType;name: Str255);
009873   INLINE $203C,$0044,midiToolNum,$A800;
009874  PROCEDURE MIDIWakeUp(refnum: INTEGER;time: LONGINT;period: LONGINT;timeProc: ProcPtr);
009875   INLINE $203C,$0048,midiToolNum,$A800;
009876  PROCEDURE MIDIRemovePort(refnum: INTEGER);
009877   INLINE $203C,$004C,midiToolNum,$A800;
009878  FUNCTION MIDIGetSync(refnum: INTEGER): INTEGER;
009879   INLINE $203C,$0050,midiToolNum,$A800;
```

```
009880  PROCEDURE MIDISetSync(refnum: INTEGER;sync: INTEGER);
009881   INLINE $203C,$0054,midiToolNum,$A800;
009882  FUNCTION MIDIGetCurTime(refnum: INTEGER): LONGINT;
009883   INLINE $203C,$0058,midiToolNum,$A800;
009884  PROCEDURE MIDISetCurTime(refnum: INTEGER;time: LONGINT);
009885   INLINE $203C,$005C,midiToolNum,$A800;
009886  PROCEDURE MIDIStartTime(refnum: INTEGER);
009887   INLINE $203C,$0060,midiToolNum,$A800;
009888  PROCEDURE MIDIStopTime(refnum: INTEGER);
009889   INLINE $203C,$0064,midiToolNum,$A800;
009890  PROCEDURE MIDIPoll(refnum: INTEGER;offsetTime: LONGINT);
009891   INLINE $203C,$0068,midiToolNum,$A800;
009892  FUNCTION MIDIWritePacket(refnum: INTEGER;packet: MIDIPacketPtr): OSErr;
009893   INLINE $203C,$006C,midiToolNum,$A800;
009894  FUNCTION MIDIWorldChanged(clientID: OSType): BOOLEAN;
009895   INLINE $203C,$0070,midiToolNum,$A800;
009896  FUNCTION MIDIGetOffsetTime(refnum: INTEGER): LONGINT;
009897   INLINE $203C,$0074,midiToolNum,$A800;
009898  PROCEDURE MIDISetOffsetTime(refnum: INTEGER;offsetTime: LONGINT);
009899   INLINE $203C,$0078,midiToolNum,$A800;
009900  FUNCTION MIDIConvertTime(srcFormat: INTEGER;dstFormat: INTEGER;time: LONGINT): LONGINT;
009901   INLINE $203C,$007C,midiToolNum,$A800;
009902  FUNCTION MIDIGetRefCon(refnum: INTEGER): LONGINT;
009903   INLINE $203C,$0080,midiToolNum,$A800;
009904  PROCEDURE MIDISetRefCon(refnum: INTEGER;refCon: LONGINT);
009905   INLINE $203C,$0084,midiToolNum,$A800;
009906  FUNCTION MIDIGetClRefCon(clientID: OSType): LONGINT;
009907   INLINE $203C,$0088,midiToolNum,$A800;
009908  PROCEDURE MIDISetClRefCon(clientID: OSType;refCon: LONGINT);
009909   INLINE $203C,$008C,midiToolNum,$A800;
009910  FUNCTION MIDIGetTCFormat(refnum: INTEGER): INTEGER;
009911   INLINE $203C,$0090,midiToolNum,$A800;
009912  PROCEDURE MIDISetTCFormat(refnum: INTEGER;format: INTEGER);
009913   INLINE $203C,$0094,midiToolNum,$A800;
009914  PROCEDURE MIDISetRunRate(refnum: INTEGER;rate: INTEGER;time: LONGINT);
009915   INLINE $203C,$0098,midiToolNum,$A800;
009916  FUNCTION MIDIGetClientIcon(clientID: OSType): Handle;
009917   INLINE $203C,$009C,midiToolNum,$A800;
009918
009919
009920  {$ENDC} { UsingMIDI }
009921
009922  {$IFC NOT UsingIncludes}
009923   END.
009924  {$ENDC}
009925
009926
009927  ### END OF FILE MIDI.p
009928
```

```
009929
009930  ##########################################################################
009931  ### FILE: Notification.p
009932  ##########################################################################
009933
009934
009935  {
009936  Created: Tuesday, September 10, 1991 at 2:10 PM
009937   Notification.p
009938   Pascal Interface to the Macintosh Libraries
009939
009940    Copyright Apple Computer, Inc. 1989-1991
009941    All rights reserved
009942  }
009943
009944
009945  {$IFC UNDEFINED UsingIncludes}
009946  {$SETC UsingIncludes := 0}
009947  {$ENDC}
009948
009949  {$IFC NOT UsingIncludes}
009950   UNIT Notification;
009951   INTERFACE
009952  {$ENDC}
009953
009954  {$IFC UNDEFINED UsingNotification}
009955  {$SETC UsingNotification := 1}
009956
009957  {$I+}
009958  {$SETC NotificationIncludes := UsingIncludes}
009959  {$SETC UsingIncludes := 1}
009960  {$IFC UNDEFINED UsingTypes}
009961  {$I $$Shell(PInterfaces)Types.p}
009962  {$ENDC}
009963  {$IFC UNDEFINED UsingOSUtils}
009964  {$I $$Shell(PInterfaces)OSUtils.p}
009965  {$ENDC}
009966  {$SETC UsingIncludes := NotificationIncludes}
009967
009968  CONST
009969  nmType = 8;
009970
009971  TYPE
009972  NMProcPtr = ProcPtr;
009973
009974  NMRecPtr = ^NMRec;
009975  NMRec = RECORD
009976   qLink: QElemPtr;       {next queue entry}
009977   qType: INTEGER;        {queue type -- ORD(nmType) = 8}
009978   nmFlags: INTEGER;      {reserved}
009979   nmPrivate: LONGINT;    {reserved}
009980   nmReserved: INTEGER;   {reserved}
009981   nmMark: INTEGER;       {item to mark in Apple menu}
009982   nmIcon: Handle;        {handle to small icon}
009983   nmSound: Handle;       {handle to sound record}
009984   nmStr: StringPtr;      {string to appear in alert}
```

```
009985   nmResp: NMProcPtr;     {pointer to response routine}
009986   nmRefCon: LONGINT;     {for application use}
009987   END;
009988
009989
009990  FUNCTION NMInstall(nmReqPtr: NMRecPtr): OSErr;
009991   INLINE $205F,$A05E,$3E80;
009992  FUNCTION NMRemove(nmReqPtr: NMRecPtr): OSErr;
009993   INLINE $205F,$A05F,$3E80;
009994
009995
009996  {$ENDC} { UsingNotification }
009997
009998  {$IFC NOT UsingIncludes}
009999   END.
010000  {$ENDC}
010001
010002
010003  ### END OF FILE Notification.p
010004
```

```
010005
010006  ############################################################################
010007  ### FILE: ObjIntf.p
010008  ############################################################################
010009
010010  {
010011      File: ObjIntf.p
010012
010013      Pascal Interface to the Macintosh Libraries
010014      Copyright Apple Computer, Inc.  1986 - 1988
010015      All rights reserved.
010016  }
010017
010018  UNIT ObjIntf;
010019
010020  INTERFACE
010021
010022  TYPE
010023      TObject = OBJECT
010024          FUNCTION  ShallowClone: TObject;
010025              {Lowest level method for copying an object; should not be overridden
010026                  except in very unusual cases.  Simply calls HandToHand to copy
010027                  the object data.}
010028          FUNCTION  Clone: TObject;
010029              {Defaults to calling ShallowClone; can be overridden to copy objects
010030                  refered to by fields.}
010031          PROCEDURE ShallowFree;
010032              {Lowest level method for freeing an object; should not be overridden
010033                  except in very unusual cases.  Simply calls DisposHandle to
010034                  free the object data.}
010035          PROCEDURE Free;
010036              {Defaults to calling ShallowFree; can be overridden to free objects
010037                  refered to by fields.}
010038          END;
010039
010040
010041  END.
010042
010043  ### END OF FILE ObjIntf.p
010044
```

```
010045
010046  ##########################################################################
010047  ### FILE: OSEvents.p
010048  ##########################################################################
010049
010050  {
010051  Created: Sunday, January 6, 1991 at 10:51 PM
010052      OSEvents.p
010053      Pascal Interface to the Macintosh Libraries
010054
010055          Copyright Apple Computer, Inc.    1985-1990
010056          All rights reserved
010057  }
010058
010059
010060  {$IFC UNDEFINED UsingIncludes}
010061  {$SETC UsingIncludes := 0}
010062  {$ENDC}
010063
010064  {$IFC NOT UsingIncludes}
010065      UNIT OSEvents;
010066      INTERFACE
010067  {$ENDC}
010068
010069  {$IFC UNDEFINED UsingOSEvents}
010070  {$SETC UsingOSEvents := 1}
010071
010072  {$I+}
010073  {$SETC OSEventsIncludes := UsingIncludes}
010074  {$SETC UsingIncludes := 1}
010075  {$IFC UNDEFINED UsingTypes}
010076  {$I $$Shell(PInterfaces)Types.p}
010077  {$ENDC}
010078  {$IFC UNDEFINED UsingEvents}
010079  {$I $$Shell(PInterfaces)Events.p}
010080  {$ENDC}
010081  {$IFC UNDEFINED UsingOSUtils}
010082  {$I $$Shell(PInterfaces)OSUtils.p}
010083  {$ENDC}
010084  {$SETC UsingIncludes := OSEventsIncludes}
010085
010086  FUNCTION PostEvent(eventNum: INTEGER;eventMsg: LONGINT): OSErr;
010087  FUNCTION PPostEvent(eventCode: INTEGER;eventMsg: LONGINT;VAR qEl: EvQElPtr): OSErr;
010088  FUNCTION OSEventAvail(mask: INTEGER;VAR theEvent: EventRecord): BOOLEAN;
010089  FUNCTION GetOSEvent(mask: INTEGER;VAR theEvent: EventRecord): BOOLEAN;
010090  PROCEDURE FlushEvents(whichMask: INTEGER;stopMask: INTEGER);
010091      INLINE $201F,$A032;
010092  PROCEDURE SetEventMask(theMask: INTEGER);
010093      INLINE $31DF,$0144;
010094  FUNCTION GetEvQHdr: QHdrPtr;
010095      INLINE $2EBC,$0000,$014A;
010096
010097
010098  {$ENDC}    { UsingOSEvents }
010099
010100  {$IFC NOT UsingIncludes}
```

```
010101      END.
010102  {$ENDC}
010103
010104
010105  ### END OF FILE OSEvents.p
010106
```

```
010107
010108  ############################################################################
010109  ### FILE: OSIntf.p
010110  ############################################################################
010111
010112  {
010113      File: OSIntf.p
010114
010115      As of MPW 3.0, interface files were reorganized to more closely
010116      match "Inside Macintosh" reference books and be more consistant
010117      from language to language.
010118
010119      Interfaces for the Operating System calls are now found in the
010120      files included below. This file is provided for compatibility
010121      with old sources.
010122
010123      Pascal Interface to the Macintosh Libraries
010124      Copyright Apple Computer, Inc. 1988
010125      All Rights Reserved
010126  }
010127
010128  {$IFC UNDEFINED UsingIncludes}
010129  {$SETC UsingIncludes := 0}
010130  {$ENDC}
010131
010132  {$IFC NOT UsingIncludes}
010133      UNIT OSIntf;
010134      INTERFACE
010135  {$ENDC}
010136
010137  {$IFC UNDEFINED UsingOSIntf}
010138  {$SETC UsingOSIntf := 1}
010139
010140  {$I+}
010141  {$SETC OSIntfIncludes := UsingIncludes}
010142  {$SETC UsingIncludes := 1}
010143  {$IFC UNDEFINED UsingEvents}
010144  {$I $$Shell(PInterfaces)Events.p}
010145  {$ENDC}
010146  {$IFC UNDEFINED UsingOSUtils}
010147  {$I $$Shell(PInterfaces)OSUtils.p}
010148  {$ENDC}
010149  {$IFC UNDEFINED UsingFiles}
010150  {$I $$Shell(PInterfaces)Files.p}
010151  {$ENDC}
010152  {$IFC UNDEFINED UsingDevices}
010153  {$I $$Shell(PInterfaces)Devices.p}
010154  {$ENDC}
010155  {$IFC UNDEFINED UsingDeskBus}
010156  {$I $$Shell(PInterfaces)DeskBus.p}
010157  {$ENDC}
010158  {$IFC UNDEFINED UsingDiskInit}
010159  {$I $$Shell(PInterfaces)DiskInit.p}
010160  {$ENDC}
010161  {$IFC UNDEFINED UsingDisks}
010162  {$I $$Shell(PInterfaces)Disks.p}
```

```
{$ENDC}
{$IFC UNDEFINED UsingErrors}
{$I $$Shell(PInterfaces)Errors.p}
{$ENDC}
{$IFC UNDEFINED UsingMemory}
{$I $$Shell(PInterfaces)Memory.p}
{$ENDC}
{$IFC UNDEFINED UsingOSEvents}
{$I $$Shell(PInterfaces)OSEvents.p}
{$ENDC}
{$IFC UNDEFINED UsingRetrace}
{$I $$Shell(PInterfaces)Retrace.p}
{$ENDC}
{$IFC UNDEFINED UsingSegLoad}
{$I $$Shell(PInterfaces)SegLoad.p}
{$ENDC}
{$IFC UNDEFINED UsingSerial}
{$I $$Shell(PInterfaces)Serial.p}
{$ENDC}
{$IFC UNDEFINED UsingShutDown}
{$I $$Shell(PInterfaces)ShutDown.p}
{$ENDC}
{$IFC UNDEFINED UsingSlots}
{$I $$Shell(PInterfaces)Slots.p}
{$ENDC}
{$IFC UNDEFINED UsingSound}
{$I $$Shell(PInterfaces)Sound.p}
{$ENDC}
{$IFC UNDEFINED UsingStart}
{$I $$Shell(PInterfaces)Start.p}
{$ENDC}
{$IFC UNDEFINED UsingTimer}
{$I $$Shell(PInterfaces)Timer.p}
{$ENDC}
{$SETC UsingIncludes := OSIntfIncludes}

{$ENDC}    { UsingOSIntf }

{$IFC NOT UsingIncludes}
    END.
{$ENDC}


### END OF FILE OSIntf.p

```

```
010208
010209  ##########################################################################
010210  ### FILE: OSUtils.p
010211  ##########################################################################
010212
010213  {
010214  Created: Sunday, January 6, 1991 at 10:51 PM
010215      OSUtils.p
010216      Pascal Interface to the Macintosh Libraries
010217
010218          Copyright Apple Computer, Inc.    1985-1990
010219          All rights reserved
010220  }
010221
010222
010223  {$IFC UNDEFINED UsingIncludes}
010224  {$SETC UsingIncludes := 0}
010225  {$ENDC}
010226
010227  {$IFC NOT UsingIncludes}
010228      UNIT OSUtils;
010229      INTERFACE
010230  {$ENDC}
010231
010232  {$IFC UNDEFINED UsingOSUtils}
010233  {$SETC UsingOSUtils := 1}
010234
010235  {$I+}
010236  {$SETC OSUtilsIncludes := UsingIncludes}
010237  {$SETC UsingIncludes := 1}
010238  {$IFC UNDEFINED UsingTypes}
010239  {$I $$Shell(PInterfaces)Types.p}
010240  {$ENDC}
010241  {$SETC UsingIncludes := OSUtilsIncludes}
010242
010243  CONST
010244  useFree = 0;
010245  useATalk = 1;
010246  useAsync = 2;
010247  useExtClk = 3;                      {Externally clocked}
010248  useMIDI = 4;
010249
010250  {*** Environs Equates ***}
010251  curSysEnvVers = 2;                  {Updated to equal latest SysEnvirons version}
010252
010253  { Machine Types }
010254  envMac = -1;
010255  envXL = -2;
010256  envMachUnknown = 0;
010257  env512KE = 1;
010258  envMacPlus = 2;
010259  envSE = 3;
010260  envMacII = 4;
010261  envMacIIx = 5;
010262  envMacIIcx = 6;
010263  envSE30 = 7;
```

```
010264  envPortable = 8;
010265  envMacIIci = 9;
010266  envMacIIfx = 11;
010267
010268  { CPU types }
010269  envCPUUnknown = 0;
010270  env68000 = 1;
010271  env68010 = 2;
010272  env68020 = 3;
010273  env68030 = 4;
010274  env68040 = 5;
010275
010276  { Keyboard types }
010277  envUnknownKbd = 0;
010278  envMacKbd = 1;
010279  envMacAndPad = 2;
010280  envMacPlusKbd = 3;
010281  envAExtendKbd = 4;
010282  envStandADBKbd = 5;
010283  envPrtblADBKbd = 6;
010284  envPrtblISOKbd = 7;
010285  envStdISOADBKbd = 8;
010286  envExtISOADBKbd = 9;
010287  false32b = 0;                       {24 bit addressing error}
010288  true32b = 1;                        {32 bit addressing error}
010289
010290  { result types for RelString Call }
010291  sortsBefore = -1;                   {first string < second string}
010292  sortsEqual = 0;                     {first string = second string}
010293  sortsAfter = 1;                     {first string > second string}
010294
010295  TYPE
010296  QTypes = (dummyType,vType,ioQType,drvQType,evType,fsQType,sIQType,dtQType);
010297
010298  TrapType = (OSTrap,ToolTrap);
010299
010300  ParamBlkType = (IOParam,FileParam,VolumeParam,CntrlParam,SlotDevParam,MultiDevParam,
010301      AccessParam,ObjParam,CopyParam,WDParam,FIDParam,CSParam,ForeignPrivParam);
010302
010303
010304  SysPPtr = ^SysParmType;
010305  SysParmType = PACKED RECORD
010306      valid: Byte;
010307      aTalkA: Byte;
010308      aTalkB: Byte;
010309      config: Byte;
010310      portA: INTEGER;
010311      portB: INTEGER;
010312      alarm: LONGINT;
010313      font: INTEGER;
010314      kbdPrint: INTEGER;
010315      volClik: INTEGER;
010316      misc: INTEGER;
010317      END;
010318
010319  {QElemPtr = ^QElem;}
```

```
010320
010321  QElemPtr = ^QElem;
010322
010323  FInfo = RECORD
010324      fdType: OSType;                 {the type of the file}
010325      fdCreator: OSType;              {file's creator}
010326      fdFlags: INTEGER;               {flags ex. hasbundle,invisible,locked, etc.}
010327      fdLocation: Point;              {file's location in folder}
010328      fdFldr: INTEGER;                {folder containing file}
010329      END;
010330
010331  VCB = RECORD
010332      qLink: QElemPtr;
010333      qType: INTEGER;
010334      vcbFlags: INTEGER;
010335      vcbSigWord: INTEGER;
010336      vcbCrDate: LONGINT;
010337      vcbLsMod: LONGINT;
010338      vcbAtrb: INTEGER;
010339      vcbNmFls: INTEGER;
010340      vcbVBMSt: INTEGER;
010341      vcbAllocPtr: INTEGER;
010342      vcbNmAlBlks: INTEGER;
010343      vcbAlBlkSiz: LONGINT;
010344      vcbClpSiz: LONGINT;
010345      vcbAlBlSt: INTEGER;
010346      vcbNxtCNID: LONGINT;
010347      vcbFreeBks: INTEGER;
010348      vcbVN: Str27;
010349      vcbDrvNum: INTEGER;
010350      vcbDRefNum: INTEGER;
010351      vcbFSID: INTEGER;
010352      vcbVRefNum: INTEGER;
010353      vcbMAdr: Ptr;
010354      vcbBufAdr: Ptr;
010355      vcbMLen: INTEGER;
010356      vcbDirIndex: INTEGER;
010357      vcbDirBlk: INTEGER;
010358      vcbVolBkUp: LONGINT;
010359      vcbVSeqNum: INTEGER;
010360      vcbWrCnt: LONGINT;
010361      vcbXTClpSiz: LONGINT;
010362      vcbCTClpSiz: LONGINT;
010363      vcbNmRtDirs: INTEGER;
010364      vcbFilCnt: LONGINT;
010365      vcbDirCnt: LONGINT;
010366      vcbFndrInfo: ARRAY [1..8] OF LONGINT;
010367      vcbVCSize: INTEGER;
010368      vcbVBMCSiz: INTEGER;
010369      vcbCtlCSiz: INTEGER;
010370      vcbXTAlBlks: INTEGER;
010371      vcbCTAlBlks: INTEGER;
010372      vcbXTRef: INTEGER;
010373      vcbCTRef: INTEGER;
010374      vcbCtlBuf: Ptr;
010375      vcbDirIDM: LONGINT;
```

```
010376      vcbOffsM: INTEGER;
010377      END;
010378
010379  DrvQElPtr = ^DrvQEl;
010380  DrvQEl = RECORD
010381      qLink: QElemPtr;
010382      qType: INTEGER;
010383      dQDrive: INTEGER;
010384      dQRefNum: INTEGER;
010385      dQFSID: INTEGER;
010386      dQDrvSz: INTEGER;
010387      dQDrvSz2: INTEGER;
010388      END;
010389
010390  ParmBlkPtr = ^ParamBlockRec;
010391  ParamBlockRec = RECORD
010392      qLink: QElemPtr;
010393      qType: INTEGER;
010394      ioTrap: INTEGER;
010395      ioCmdAddr: Ptr;
010396      ioCompletion: ProcPtr;
010397      ioResult: OSErr;
010398      ioNamePtr: StringPtr;
010399      ioVRefNum: INTEGER;
010400      CASE ParamBlkType OF
010401        IOParam:
010402          (ioRefNum: INTEGER;
010403          ioVersNum: SignedByte;
010404          ioPermssn: SignedByte;
010405          ioMisc: Ptr;
010406          ioBuffer: Ptr;
010407          ioReqCount: LONGINT;
010408          ioActCount: LONGINT;
010409          ioPosMode: INTEGER;
010410          ioPosOffset: LONGINT);
010411        FileParam:
010412          (ioFRefNum: INTEGER;
010413          ioFVersNum: SignedByte;
010414          filler1: SignedByte;
010415          ioFDirIndex: INTEGER;
010416          ioFlAttrib: SignedByte;
010417          ioFlVersNum: SignedByte;
010418          ioFlFndrInfo: FInfo;
010419          ioFlNum: LONGINT;
010420          ioFlStBlk: INTEGER;
010421          ioFlLgLen: LONGINT;
010422          ioFlPyLen: LONGINT;
010423          ioFlRStBlk: INTEGER;
010424          ioFlRLgLen: LONGINT;
010425          ioFlRPyLen: LONGINT;
010426          ioFlCrDat: LONGINT;
010427          ioFlMdDat: LONGINT);
010428        VolumeParam:
010429          (filler2: LONGINT;
010430          ioVolIndex: INTEGER;
010431          ioVCrDate: LONGINT;
```

```
010432          ioVLsBkUp: LONGINT;
010433          ioVAtrb: INTEGER;
010434          ioVNmFls: INTEGER;
010435          ioVDirSt: INTEGER;
010436          ioVBlLn: INTEGER;
010437          ioVNmAlBlks: INTEGER;
010438          ioVAlBlkSiz: LONGINT;
010439          ioVClpSiz: LONGINT;
010440          ioAlBlSt: INTEGER;
010441          ioVNxtFNum: LONGINT;
010442          ioVFrBlk: INTEGER);
010443        CntrlParam:
010444          (ioCRefNum: INTEGER;
010445          csCode: INTEGER;
010446          csParam: ARRAY [0..10] OF INTEGER);
010447        SlotDevParam:
010448          (filler3: LONGINT;
010449          ioMix: Ptr;
010450          ioFlags: INTEGER;
010451          ioSlot: SignedByte;
010452          ioID: SignedByte);
010453        MultiDevParam:
010454          (filler4: LONGINT;
010455          ioMMix: Ptr;
010456          ioMFlags: INTEGER;
010457          ioSEBlkPtr: Ptr);
010458      END;
010459
010460  EvQElPtr = ^EvQEl;
010461  EvQEl = RECORD
010462      qLink: QElemPtr;
010463      qType: INTEGER;
010464      evtQWhat: INTEGER;              {this part is identical to the EventRecord as...}
010465      evtQMessage: LONGINT;           {defined in ToolIntf}
010466      evtQWhen: LONGINT;
010467      evtQWhere: Point;
010468      evtQModifiers: INTEGER;
010469      END;
010470
010471  VBLTask = RECORD
010472      qLink: QElemPtr;
010473      qType: INTEGER;
010474      vblAddr: ProcPtr;
010475      vblCount: INTEGER;
010476      vblPhase: INTEGER;
010477      END;
010478
010479  DeferredTask = RECORD
010480      qLink: QElemPtr;                {next queue entry}
010481      qType: INTEGER;                 {queue type}
010482      dtFlags: INTEGER;               {reserved}
010483      dtAddr: ProcPtr;                {pointer to task}
010484      dtParm: LONGINT;                {optional parameter}
010485      dtReserved: LONGINT;            {reserved--should be 0}
010486      END;
010487
```

```
010488  QElem = RECORD
010489      CASE QTypes OF
010490        dtQType:
010491          (dtQElem: DeferredTask);    {deferred}
010492        vType:
010493          (vblQElem: VBLTask);        {vertical blanking}
010494        ioQType:
010495          (ioQElem: ParamBlockRec);   {I/O parameter block}
010496        drvQType:
010497          (drvQElem: DrvQEl);         {drive}
010498        evType:
010499          (evQElem: EvQEl);           {event}
010500        fsQType:
010501          (vcbQElem: VCB);            {volume control block}
010502      END;
010503
010504  QHdrPtr = ^QHdr;
010505  QHdr = RECORD
010506      qFlags: INTEGER;
010507      qHead: QElemPtr;
010508      qTail: QElemPtr;
010509      END;
010510
010511  DateTimeRec = RECORD
010512      year: INTEGER;
010513      month: INTEGER;
010514      day: INTEGER;
010515      hour: INTEGER;
010516      minute: INTEGER;
010517      second: INTEGER;
010518      dayOfWeek: INTEGER;
010519      END;
010520
010521  SysEnvRec = RECORD
010522      environsVersion: INTEGER;
010523      machineType: INTEGER;
010524      systemVersion: INTEGER;
010525      processor: INTEGER;
010526      hasFPU: BOOLEAN;
010527      hasColorQD: BOOLEAN;
010528      keyBoardType: INTEGER;
010529      atDrvrVersNum: INTEGER;
010530      sysVRefNum: INTEGER;
010531      END;
010532
010533
010534  FUNCTION GetSysPPtr: SysPPtr;
010535      INLINE $2EBC,$0000,$01F8;
010536  PROCEDURE SysBeep(duration: INTEGER);
010537      INLINE $A9C8;
010538  FUNCTION KeyTrans(transData: Ptr;keycode: INTEGER;VAR state: LONGINT): LONGINT;
010539      INLINE $A9C3;
010540  FUNCTION DTInstall(dtTaskPtr: QElemPtr): OSErr;
010541  FUNCTION GetMMUMode: SignedByte;
010542  PROCEDURE SwapMMUMode(VAR mode: SignedByte);
010543  FUNCTION SysEnvirons(versionRequested: INTEGER;VAR theWorld: SysEnvRec): OSErr;
```

```
010544  FUNCTION ReadDateTime(VAR time: LONGINT): OSErr;
010545  PROCEDURE GetDateTime(VAR secs: LONGINT);
010546  FUNCTION SetDateTime(time: LONGINT): OSErr;
010547  PROCEDURE SetTime(d: DateTimeRec);
010548  PROCEDURE GetTime(VAR d: DateTimeRec);
010549  PROCEDURE Date2Secs(d: DateTimeRec;VAR secs: LONGINT);
010550  PROCEDURE Secs2Date(secs: LONGINT;VAR d: DateTimeRec);
010551  PROCEDURE Delay(numTicks: LONGINT;VAR finalTicks: LONGINT);
010552  FUNCTION GetTrapAddress(trapNum: INTEGER): LONGINT;
010553  PROCEDURE SetTrapAddress(trapAddr: LONGINT;trapNum: INTEGER);
010554  FUNCTION NGetTrapAddress(trapNum: INTEGER;tTyp: TrapType): LONGINT;
010555  PROCEDURE NSetTrapAddress(trapAddr: LONGINT;trapNum: INTEGER;tTyp: TrapType);
010556  FUNCTION GetOSTrapAddress(trapNum: INTEGER): LONGINT;
010557  PROCEDURE SetOSTrapAddress(trapAddr: LONGINT;trapNum: INTEGER);
010558  FUNCTION GetToolTrapAddress(trapNum: INTEGER): LONGINT;
010559  PROCEDURE SetToolTrapAddress(trapAddr: LONGINT;trapNum: INTEGER);
010560  FUNCTION GetToolboxTrapAddress(trapNum: INTEGER): LONGINT;
010561  PROCEDURE SetToolboxTrapAddress(trapAddr: LONGINT;trapNum: INTEGER);
010562  FUNCTION WriteParam: OSErr;
010563  FUNCTION EqualString(str1: Str255;str2: Str255;caseSens: BOOLEAN;diacSens: BOOLEAN): BOOLEAN;
010564  PROCEDURE UprString(VAR theString: Str255;diacSens: BOOLEAN);
010565  PROCEDURE Enqueue(qElement: QElemPtr;qHeader: QHdrPtr);
010566  FUNCTION Dequeue(qElement: QElemPtr;qHeader: QHdrPtr): OSErr;
010567  FUNCTION SetCurrentA5: LONGINT;
010568      INLINE $2E8D,$2A78,$0904;
010569  FUNCTION SetA5(newA5: LONGINT): LONGINT;
010570      INLINE $2F4D,$0004,$2A5F;
010571  PROCEDURE Environs(VAR rom: INTEGER;VAR machine: INTEGER);
010572  FUNCTION RelString(str1: Str255;str2: Str255;caseSens: BOOLEAN;diacSens: BOOLEAN): INTEGER;
010573  FUNCTION HandToHand(VAR theHndl: Handle): OSErr;
010574  FUNCTION PtrToXHand(srcPtr: Ptr;dstHndl: Handle;size: LONGINT): OSErr;
010575  FUNCTION PtrToHand(srcPtr: Ptr;VAR dstHndl: Handle;size: LONGINT): OSErr;
010576  FUNCTION HandAndHand(hand1: Handle;hand2: Handle): OSErr;
010577  FUNCTION PtrAndHand(ptr1: Ptr;hand2: Handle;size: LONGINT): OSErr;
010578  FUNCTION InitUtil: OSErr;
010579      INLINE $A03F,$3E80;
010580  FUNCTION SwapInstructionCache(cacheEnable: BOOLEAN): BOOLEAN;
010581  PROCEDURE FlushInstructionCache;
010582  FUNCTION SwapDataCache(cacheEnable: BOOLEAN): BOOLEAN;
010583  PROCEDURE FlushDataCache;
010584
010585
010586  {$ENDC}    { UsingOSUtils }
010587
010588  {$IFC NOT UsingIncludes}
010589      END.
010590  {$ENDC}
010591
010592
010593  ### END OF FILE OSUtils.p
010594
```

```
010595
010596  ##########################################################################
010597  ### FILE: Packages.p
010598  ##########################################################################
010599
010600
010601  {
010602  Created: Sunday, September 15, 1991 at 11:56 PM
010603   Packages.p
010604   Pascal Interface to the Macintosh Libraries
010605
010606    Copyright Apple Computer, Inc. 1985-1991
010607    All rights reserved
010608  }
010609
010610
010611  {$IFC UNDEFINED UsingIncludes}
010612  {$SETC UsingIncludes := 0}
010613  {$ENDC}
010614
010615  {$IFC NOT UsingIncludes}
010616   UNIT Packages;
010617   INTERFACE
010618  {$ENDC}
010619
010620  {$IFC UNDEFINED UsingPackages}
010621  {$SETC UsingPackages := 1}
010622
010623  {$I+}
010624  {$SETC PackagesIncludes := UsingIncludes}
010625  {$SETC UsingIncludes := 1}
010626  {$IFC UNDEFINED UsingTypes}
010627  {$I $$Shell(PInterfaces)Types.p}
010628  {$ENDC}
010629  {$IFC UNDEFINED UsingStandardFile}
010630  {$I $$Shell(PInterfaces)StandardFile.p}
010631  {$ENDC}
010632  {$IFC UNDEFINED UsingScript}
010633  {$I $$Shell(PInterfaces)Script.p}
010634  {$ENDC}
010635  {$SETC UsingIncludes := PackagesIncludes}
010636
010637  CONST
010638  listMgr = 0;                            {list manager}
010639  dskInit = 2;                            {Disk Initializaton}
010640  stdFile = 3;                            {Standard File}
010641  flPoint = 4;                            {Floating-Point Arithmetic}
010642  trFunc = 5;                             {Transcendental Functions}
010643  intUtil = 6;                            {International Utilities}
010644  bdConv = 7;                             {Binary/Decimal Conversion}
010645  editionMgr = 11;                        {Edition Manager}
010646  currSymLead = 16;
010647  currNegSym = 32;
010648  currTrailingZ = 64;
010649  currLeadingZ = 128;
010650  zeroCycle = 1;                          {0:00 AM/PM format}
```

```
010651  longDay = 0;                           {day of the month}
010652  longWeek = 1;                          {day of the week}
010653  longMonth = 2;                         {month of the year}
010654  longYear = 3;                          {year}
010655  supDay = 1;                            {suppress day of month}
010656  supWeek = 2;                           {suppress day of week}
010657  supMonth = 4;                          {suppress month}
010658  supYear = 8;                           {suppress year}
010659  dayLdingZ = 32;
010660  mntLdingZ = 64;
010661  century = 128;
010662  secLeadingZ = 32;
010663  minLeadingZ = 64;
010664  hrLeadingZ = 128;
010665
010666  { Date Orders }
010667  mdy = 0;
010668  dmy = 1;
010669  ymd = 2;
010670  myd = 3;
010671  dym = 4;
010672  ydm = 5;
010673
010674
010675  { Regional version codes }
010676  verUS = 0;
010677  verFrance = 1;
010678  verBritain = 2;
010679  verGermany = 3;
010680  verItaly = 4;
010681  verNetherlands = 5;
010682  verFrBelgiumLux = 6;                   { French for Belgium & Luxembourg }
010683  verSweden = 7;
010684  verSpain = 8;
010685  verDenmark = 9;
010686  verPortugal = 10;
010687  verFrCanada = 11;
010688  verNorway = 12;
010689  verIsrael = 13;
010690  verJapan = 14;
010691  verAustralia = 15;
010692  verArabic = 16;                        { synonym for verArabia }
010693  verFinland = 17;
010694  verFrSwiss = 18;                       { French Swiss }
010695  verGrSwiss = 19;                       { German Swiss }
010696  verGreece = 20;
010697  verIceland = 21;
010698  verMalta = 22;
010699  verCyprus = 23;
010700  verTurkey = 24;
010701  verYugoCroatian = 25;                  { Croatian system for Yugoslavia }
010702  verIndiaHindi = 33;                    { Hindi system for India }
010703  verPakistan = 34;
010704  verLithuania = 41;
010705  verPoland = 42;
010706  verHungary = 43;
```

```
010707  verEstonia = 44;
010708  verLatvia = 45;
010709  verLapland = 46;
010710  verFaeroeIsl = 47;
010711  verIran = 48;
010712  verRussia = 49;
010713  verIreland = 50;                        { English-language version for Ireland }
010714  verKorea = 51;
010715  verChina = 52;
010716  verTaiwan = 53;
010717  verThailand = 54;
010718  minCountry = verUS;
010719  maxCountry = verThailand;
010720
010721  {Obsolete region code names, kept for backward compatibility}
010722  verBelgiumLux = 6;                      { (use verFrBelgiumLux instead, less ambiguous) }
010723  verArabia = 16;
010724  verYugoslavia = 25;                     { (use verYugoCroatian instead, less ambiguous) }
010725  verIndia = 33;                          { (use verIndiaHindi instead, less ambiguous) }
010726
010727  { Special script code values for International Utilities }
010728  iuSystemScript = -1;                    { system script }
010729  iuCurrentScript = -2;                   { current script (for font of grafPort) }
010730
010731  { Special language code values for International Utilities }
010732  iuSystemCurLang = -2;                   { current (itlbLang) lang for system script }
010733  iuSystemDefLang = -3;                   { default (table) lang for system script }
010734  iuCurrentCurLang = -4;                  { current (itlbLang) lang for current script }
010735  iuCurrentDefLang = -5;                  { default lang for current script }
010736  iuScriptCurLang = -6;                   { current (itlbLang) lang for specified script }
010737  iuScriptDefLang = -7;                   { default language for a specified script }
010738
010739  { Table selectors for GetItlTable }
010740  iuWordSelectTable = 0;                  { get word select break table from 'itl2' }
010741  iuWordWrapTable = 1;                    { get word wrap break table from 'itl2' }
010742  iuNumberPartsTable = 2;                 { get default number parts table from 'itl4' }
010743  iuUnTokenTable = 3;                     { get unToken table from 'itl4' }
010744  iuWhiteSpaceList = 4;                   { get white space list from 'itl4' }
010745
010746  TYPE
010747  DateForm = (shortDate,longDate,abbrevDate);
010748
010749
010750  Intl0Ptr = ^Intl0Rec;
010751  Intl0Hndl = ^Intl0Ptr;
010752  Intl0Rec = PACKED RECORD
010753   decimalPt: CHAR;                       {decimal point character}
010754   thousSep: CHAR;                        {thousands separator character}
010755   listSep: CHAR;                         {list separator character}
010756   currSym1: CHAR;                        {currency symbol}
010757   currSym2: CHAR;
010758   currSym3: CHAR;
010759   currFmt: Byte;                         {currency format flags}
010760   dateOrder: Byte;                       {order of short date elements: mdy, dmy, etc.}
010761   shrtDateFmt: Byte;                     {format flags for each short date element}
010762   dateSep: CHAR;                         {date separator character}
```

```
010763   timeCycle: Byte;                       {specifies time cycle: 0..23, 1..12, or 0..11}
010764   timeFmt: Byte;                         {format flags for each time element}
010765   mornStr: PACKED ARRAY [1..4] OF CHAR;  {trailing string for AM if 12-hour cycle}
010766   eveStr: PACKED ARRAY [1..4] OF CHAR;   {trailing string for PM if 12-hour cycle}
010767   timeSep: CHAR;                         {time separator character}
010768   time1Suff: CHAR;                       {trailing string for AM if 24-hour cycle}
010769   time2Suff: CHAR;
010770   time3Suff: CHAR;
010771   time4Suff: CHAR;
010772   time5Suff: CHAR;                       {trailing string for PM if 24-hour cycle}
010773   time6Suff: CHAR;
010774   time7Suff: CHAR;
010775   time8Suff: CHAR;
010776   metricSys: Byte;                       {255 if metric, 0 if inches etc.}
010777   intl0Vers: INTEGER;                    {region code (hi byte) and version (lo byte)}
010778   END;
010779
010780  Intl1Ptr = ^Intl1Rec;
010781  Intl1Hndl = ^Intl1Ptr;
010782  Intl1Rec = PACKED RECORD
010783   days: ARRAY [1..7] OF Str15;           {day names}
010784   months: ARRAY [1..12] OF Str15;        {month names}
010785   suppressDay: Byte;                     {255 for no day, or flags to suppress any element}
010786   lngDateFmt: Byte;                      {order of long date elements}
010787   dayLeading0: Byte;                     {255 for leading 0 in day number}
010788   abbrLen: Byte;                         {length for abbreviating names}
010789   st0: PACKED ARRAY [1..4] OF CHAR;      {separator strings for long date format}
010790   st1: PACKED ARRAY [1..4] OF CHAR;
010791   st2: PACKED ARRAY [1..4] OF CHAR;
010792   st3: PACKED ARRAY [1..4] OF CHAR;
010793   st4: PACKED ARRAY [1..4] OF CHAR;
010794   intl1Vers: INTEGER;                    {region code (hi byte) and version (lo byte)}
010795   localRtn: ARRAY [0..0] OF INTEGER;     {now a flag for opt extension}
010796   END;
010797
010798
010799  PROCEDURE InitPack(packID: INTEGER);
010800   INLINE $A9E5;
010801  PROCEDURE InitAllPacks;
010802   INLINE $A9E6;
010803
010804  FUNCTION IUGetIntl(theID: INTEGER): Handle;
010805   INLINE $3F3C,$0006,$A9ED;
010806  PROCEDURE IUSetIntl(refNum: INTEGER;theID: INTEGER;intlHandle: Handle);
010807   INLINE $3F3C,$0008,$A9ED;
010808  PROCEDURE IUDateString(dateTime: LONGINT;longFlag: DateForm;VAR result: Str255);
010809   INLINE $4267,$A9ED;
010810  PROCEDURE IUDatePString(dateTime: LONGINT;longFlag: DateForm;VAR result: Str255;
010811   intlHandle: Handle);
010812   INLINE $3F3C,$000E,$A9ED;
010813  PROCEDURE IUTimeString(dateTime: LONGINT;wantSeconds: BOOLEAN;VAR result: Str255);
010814   INLINE $3F3C,$0002,$A9ED;
010815  PROCEDURE IUTimePString(dateTime: LONGINT;wantSeconds: BOOLEAN;VAR result: Str255;
010816   intlHandle: Handle);
010817   INLINE $3F3C,$0010,$A9ED;
010818  FUNCTION IUMetric: BOOLEAN;
```

```
010819   INLINE $3F3C,$0004,$A9ED;
010820
010821  FUNCTION IUMagString(aPtr: Ptr;bPtr: Ptr;aLen: INTEGER;bLen: INTEGER): INTEGER;
010822   INLINE $3F3C,$000A,$A9ED;
010823  FUNCTION IUMagIDString(aPtr: Ptr;bPtr: Ptr;aLen: INTEGER;bLen: INTEGER): INTEGER;
010824   INLINE $3F3C,$000C,$A9ED;
010825  FUNCTION IUCompString(aStr: Str255;bStr: Str255): INTEGER;
010826  FUNCTION IUEqualString(aStr: Str255;bStr: Str255): INTEGER;
010827
010828  PROCEDURE StringToNum(theString: Str255;VAR theNum: LONGINT);
010829  PROCEDURE NumToString(theNum: LONGINT;VAR theString: Str255);
010830
010831  PROCEDURE IULDateString(VAR dateTime: LongDateTime;longFlag: DateForm;VAR result: Str255;
010832   intlHandle: Handle);
010833   INLINE $3F3C,$0014,$A9ED;
010834  PROCEDURE IULTimeString(VAR dateTime: LongDateTime;wantSeconds: BOOLEAN;
010835   VAR result: Str255;intlHandle: Handle);
010836   INLINE $3F3C,$0016,$A9ED;
010837  PROCEDURE IUClearCache;
010838   INLINE $3F3C,$0018,$A9ED;
010839  FUNCTION IUMagPString(aPtr: Ptr;bPtr: Ptr;aLen: INTEGER;bLen: INTEGER;itl2Handle: Handle): INTEGER;
010840   INLINE $3F3C,$001A,$A9ED;
010841  FUNCTION IUMagIDPString(aPtr: Ptr;bPtr: Ptr;aLen: INTEGER;bLen: INTEGER;
010842   itl2Handle: Handle): INTEGER;
010843   INLINE $3F3C,$001C,$A9ED;
010844  FUNCTION IUCompPString(aStr: Str255;bStr: Str255;itl2Handle: Handle): INTEGER;
010845  FUNCTION IUEqualPString(aStr: Str255;bStr: Str255;itl2Handle: Handle): INTEGER;
010846  FUNCTION IUScriptOrder(script1: ScriptCode;script2: ScriptCode): INTEGER;
010847   INLINE $3F3C,$001E,$A9ED;
010848  FUNCTION IULangOrder(language1: LangCode;language2: LangCode): INTEGER;
010849   INLINE $3F3C,$0020,$A9ED;
010850  FUNCTION IUTextOrder(aPtr: Ptr;bPtr: Ptr;aLen: INTEGER;bLen: INTEGER;aScript: ScriptCode;
010851   bScript: ScriptCode;aLang: LangCode;bLang: LangCode): INTEGER;
010852   INLINE $3F3C,$0022,$A9ED;
010853  FUNCTION IUStringOrder(aStr: Str255;bStr: Str255;aScript: ScriptCode;bScript: ScriptCode;
010854   aLang: LangCode;bLang: LangCode): INTEGER;
010855  PROCEDURE IUGetItlTable(script: ScriptCode;tableCode: INTEGER;VAR itlHandle: Handle;
010856   VAR offset: LONGINT;VAR length: LONGINT);
010857   INLINE $3F3C,$0024,$A9ED;
010858
010859
010860  {$ENDC} { UsingPackages }
010861
010862  {$IFC NOT UsingIncludes}
010863   END.
010864  {$ENDC}
010865
010866
010867  ### END OF FILE Packages.p
010868
```

```
010869
010870  ############################################################################
010871  ### FILE: PackIntf.p
010872  ############################################################################
010873
010874  {
010875      File: PackIntf.p
010876
010877      As of MPW 3.0, interface files were reorganized to more closely
010878      match "Inside Macintosh" reference books and be more consistant
010879      from language to language.
010880
010881      Interfaces for the Package Manager are now found in Packages.p.
010882      This file, which includes Packages.p, is provided for compatibility
010883      with old sources.
010884
010885      Pascal Interface to the Macintosh Libraries
010886      Copyright Apple Computer, Inc. 1988
010887      All Rights Reserved
010888  }
010889
010890  {$IFC UNDEFINED UsingIncludes}
010891  {$SETC UsingIncludes := 0}
010892  {$ENDC}
010893
010894  {$IFC NOT UsingIncludes}
010895      UNIT PackIntf;
010896      INTERFACE
010897  {$ENDC}
010898
010899  {$IFC UNDEFINED UsingPackIntf}
010900  {$SETC UsingPackIntf := 1}
010901
010902  {$I+}
010903  {$SETC PackIntfIncludes := UsingIncludes}
010904  {$SETC UsingIncludes := 1}
010905  {$IFC UNDEFINED UsingPackages}
010906  {$I $$Shell(PInterfaces)Packages.p}
010907  {$ENDC}
010908  {$SETC UsingIncludes := PackIntfIncludes}
010909
010910  {$ENDC}    { UsingPackIntf }
010911
010912  {$IFC NOT UsingIncludes}
010913      END.
010914  {$ENDC}
010915
010916
010917  ### END OF FILE PackIntf.p
010918
```

```
010919
010920  ############################################################################
010921  ### FILE: PaletteMgr.p
010922  ############################################################################
010923
010924  {
010925      File: PaletteMgr.p
010926
010927      As of MPW 3.0, interface files were reorganized to more closely
010928      match "Inside Macintosh" reference books and be more consistant
010929      from language to language.
010930
010931      Interfaces for the Palette Manager are now found in Palettes.p.
010932      This file, which includes Palettes.p, is provided for compatibility
010933      with old sources.
010934
010935      Pascal Interface to the Macintosh Libraries
010936      Copyright Apple Computer, Inc. 1988
010937      All Rights Reserved
010938  }
010939
010940  {$IFC UNDEFINED UsingIncludes}
010941  {$SETC UsingIncludes := 0}
010942  {$ENDC}
010943
010944  {$IFC NOT UsingIncludes}
010945      UNIT PaletteMgr;
010946      INTERFACE
010947  {$ENDC}
010948
010949  {$IFC UNDEFINED UsingPaletteMgr}
010950  {$SETC UsingPaletteMgr := 1}
010951
010952  {$I+}
010953  {$SETC PaletteMgrIncludes := UsingIncludes}
010954  {$SETC UsingIncludes := 1}
010955  {$IFC UNDEFINED UsingPalettes}
010956  {$I $$Shell(PInterfaces)Palettes.p}
010957  {$ENDC}
010958  {$SETC UsingIncludes := PaletteMgrIncludes}
010959
010960  {$ENDC}    { UsingPaletteMgr }
010961
010962  {$IFC NOT UsingIncludes}
010963      END.
010964  {$ENDC}
010965
010966
010967  ### END OF FILE PaletteMgr.p
010968
```

```
010969
010970  ############################################################################
010971  ### FILE: Palettes.p
010972  ############################################################################
010973
010974
010975  {
010976  Created: Monday, September 16, 1991 at 12:00 AM
010977   Palettes.p
010978   Pascal Interface to the Macintosh Libraries
010979
010980    Copyright Apple Computer, Inc. 1987-1991
010981    All rights reserved
010982  }
010983
010984
010985  {$IFC UNDEFINED UsingIncludes}
010986  {$SETC UsingIncludes := 0}
010987  {$ENDC}
010988
010989  {$IFC NOT UsingIncludes}
010990   UNIT Palettes;
010991   INTERFACE
010992  {$ENDC}
010993
010994  {$IFC UNDEFINED UsingPalettes}
010995  {$SETC UsingPalettes := 1}
010996
010997  {$I+}
010998  {$SETC PalettesIncludes := UsingIncludes}
010999  {$SETC UsingIncludes := 1}
011000  {$IFC UNDEFINED UsingQuickdraw}
011001  {$I $$Shell(PInterfaces)Quickdraw.p}
011002  {$ENDC}
011003  {$IFC UNDEFINED UsingWindows}
011004  {$I $$Shell(PInterfaces)Windows.p}
011005  {$ENDC}
011006  {$SETC UsingIncludes := PalettesIncludes}
011007
011008  CONST
011009  pmCourteous = 0;                        {Record use of color on each device touched.}
011010  pmTolerant = $0002;                     {render ciRGB if ciTolerance is exceeded by best match.}
011011  pmAnimated = $0004;                     {reserve an index on each device touched and render ciRGB.}
011012  pmExplicit = $0008;                     {no reserve, no render, no record; stuff index into port.}
011013
011014  pmWhite = $0010;
011015  pmBlack = $0020;
011016
011017  pmInhibitG2 = $0100;
011018  pmInhibitC2 = $0200;
011019  pmInhibitG4 = $0400;
011020  pmInhibitC4 = $0800;
011021  pmInhibitG8 = $1000;
011022  pmInhibitC8 = $2000;
011023
011024
```

```
011025  { NSetPalette Update Constants }
011026  pmNoUpdates = $8000;                        {no updates}
011027  pmBkUpdates = $A000;                        {background updates only}
011028  pmFgUpdates = $C000;                        {foreground updates only}
011029  pmAllUpdates = $E000;                       {all updates}
011030
011031  TYPE
011032  ColorInfo = RECORD
011033   ciRGB: RGBColor;                           {true RGB values}
011034   ciUsage: INTEGER;                          {color usage}
011035   ciTolerance: INTEGER;                      {tolerance value}
011036   ciDataFields: ARRAY [0..2] OF INTEGER;     {private fields}
011037   END;
011038
011039  PalettePtr = ^Palette;
011040  PaletteHandle = ^PalettePtr;
011041  Palette = RECORD
011042   pmEntries: INTEGER;                        {entries in pmTable}
011043   pmDataFields: ARRAY [0..6] OF INTEGER;     {private fields}
011044   pmInfo: ARRAY [0..0] OF ColorInfo;
011045   END;
011046
011047
011048  PROCEDURE InitPalettes;
011049   INLINE $AA90;
011050  FUNCTION NewPalette(entries: INTEGER;srcColors: CTabHandle;srcUsage: INTEGER;
011051   srcTolerance: INTEGER): PaletteHandle;
011052   INLINE $AA91;
011053  FUNCTION GetNewPalette(PaletteID: INTEGER): PaletteHandle;
011054   INLINE $AA92;
011055  PROCEDURE DisposePalette(srcPalette: PaletteHandle);
011056   INLINE $AA93;
011057  PROCEDURE ActivatePalette(srcWindow: WindowPtr);
011058   INLINE $AA94;
011059  PROCEDURE SetPalette(dstWindow: WindowPtr;srcPalette: PaletteHandle;cUpdates: BOOLEAN);
011060   INLINE $AA95;
011061  PROCEDURE NSetPalette(dstWindow: WindowPtr;srcPalette: PaletteHandle;nCUpdates: INTEGER);
011062   INLINE $AA95;
011063  FUNCTION GetPalette(srcWindow: WindowPtr): PaletteHandle;
011064   INLINE $AA96;
011065  PROCEDURE CopyPalette(srcPalette: PaletteHandle;dstPalette: PaletteHandle;
011066   srcEntry: INTEGER;dstEntry: INTEGER;dstLength: INTEGER);
011067   INLINE $AAA1;
011068  PROCEDURE PmForeColor(dstEntry: INTEGER);
011069   INLINE $AA97;
011070  PROCEDURE PmBackColor(dstEntry: INTEGER);
011071   INLINE $AA98;
011072  PROCEDURE AnimateEntry(dstWindow: WindowPtr;dstEntry: INTEGER;srcRGB: RGBColor);
011073   INLINE $AA99;
011074  PROCEDURE AnimatePalette(dstWindow: WindowPtr;srcCTab: CTabHandle;srcIndex: INTEGER;
011075   dstEntry: INTEGER;dstLength: INTEGER);
011076   INLINE $AA9A;
011077  PROCEDURE GetEntryColor(srcPalette: PaletteHandle;srcEntry: INTEGER;VAR dstRGB: RGBColor);
011078   INLINE $AA9B;
011079  PROCEDURE SetEntryColor(dstPalette: PaletteHandle;dstEntry: INTEGER;srcRGB: RGBColor);
011080   INLINE $AA9C;
```

```
011081  PROCEDURE GetEntryUsage(srcPalette: PaletteHandle;srcEntry: INTEGER;VAR dstUsage: INTEGER;
011082   VAR dstTolerance: INTEGER);
011083   INLINE $AA9D;
011084  PROCEDURE SetEntryUsage(dstPalette: PaletteHandle;dstEntry: INTEGER;srcUsage: INTEGER;
011085   srcTolerance: INTEGER);
011086   INLINE $AA9E;
011087  PROCEDURE CTab2Palette(srcCTab: CTabHandle;dstPalette: PaletteHandle;srcUsage: INTEGER;
011088   srcTolerance: INTEGER);
011089   INLINE $AA9F;
011090  PROCEDURE Palette2CTab(srcPalette: PaletteHandle;dstCTab: CTabHandle);
011091   INLINE $AAA0;
011092  FUNCTION Entry2Index(entry: INTEGER): LONGINT;
011093   INLINE $7000,$AAA2;
011094  PROCEDURE RestoreDeviceClut(gd: GDHandle);
011095   INLINE $7002,$AAA2;
011096  PROCEDURE ResizePalette(p: PaletteHandle;size: INTEGER);
011097   INLINE $7003,$AAA2;
011098  PROCEDURE SaveFore(VAR c: ColorSpec);
011099   INLINE $303C,$040D,$AAA2;
011100  PROCEDURE SaveBack(VAR c: ColorSpec);
011101   INLINE $303C,$040E,$AAA2;
011102  PROCEDURE RestoreFore(c: ColorSpec);
011103   INLINE $303C,$040F,$AAA2;
011104  PROCEDURE RestoreBack(c: ColorSpec);
011105   INLINE $303C,$0410,$AAA2;
011106  FUNCTION SetDepth(gd: GDHandle;depth: INTEGER;whichFlags: INTEGER;flags: INTEGER): OSErr;
011107   INLINE $303C,$0A13,$AAA2;
011108  FUNCTION HasDepth(gd: GDHandle;depth: INTEGER;whichFlags: INTEGER;flags: INTEGER): INTEGER;
011109   INLINE $303C,$0A14,$AAA2;
011110  FUNCTION PMgrVersion: INTEGER;
011111   INLINE $7015,$AAA2;
011112  PROCEDURE SetPaletteUpdates(p: PaletteHandle;updates: INTEGER);
011113   INLINE $303C,$0616,$AAA2;
011114  FUNCTION GetPaletteUpdates(p: PaletteHandle): INTEGER;
011115   INLINE $303C,$0417,$AAA2;
011116  FUNCTION GetGray(device: GDHandle;backGround: RGBColor;VAR foreGround: RGBColor): BOOLEAN;
011117   INLINE $303C,$0C19,$AAA2;
011118
011119
011120  {$ENDC} { UsingPalettes }
011121
011122  {$IFC NOT UsingIncludes}
011123   END.
011124  {$ENDC}
011125
011126
011127  ### END OF FILE Palettes.p
011128
```

```
011129
011130  ############################################################################
011131  ### FILE: PasLibIntf.p
011132  ############################################################################
011133
011134
011135  {
011136  Created: Monday, January 22, 1990 at 9:18 PM
011137      PasLibIntf.p
011138      Pascal Interface to the Macintosh Libraries
011139
011140      Copyright Apple Computer, Inc.  1986-1991
011141      All rights reserved
011142
011143
011144      Interface to the Pascal I/O and Memory Manager Library.
011145      Built-in procedure and function declarations are marked with
011146      the (* *) comment characters
011147
011148  }
011149
011150
011151  {$IFC UNDEFINED UsingIncludes}
011152  {$SETC UsingIncludes := 0}
011153  {$ENDC}
011154
011155  {$IFC NOT UsingIncludes}
011156      UNIT PASLIBIntf;
011157      INTERFACE
011158  {$ENDC}
011159
011160  {$IFC UNDEFINED UsingPASLIBINTF}
011161  {$SETC UsingPASLIBINTF := 1}
011162
011163  {$I+}
011164  {$SETC PASLIBINTFIncludes := UsingIncludes}
011165  {$SETC UsingIncludes := 1}
011166  {$IFC UNDEFINED UsingTypes}
011167  {$I $$Shell(PInterfaces)Types.p}
011168  {$ENDC}
011169  {$SETC UsingIncludes := PASLIBINTFIncludes}
011170
011171      TYPE
011172        PASCALPOINTER = ^INTEGER; { Universal POINTER type }
011173        PASCALFILE = FILE; { Universal FILE type }
011174  (*
011175   *    PASCALBLOCK =    { Universal block of chars }
011176   *          PACKED ARRAY [0..511] OF CHAR;
011177   *)
011178
011179      CONST
011180        { <StdIO.h> PLSetVBuf styles }
011181        _IOFBF = $00; { File buffering }
011182        _IOLBF = $40; { Line buffering }
011183        _IONBF = $04; { No buffering }
011184
```

```
011185  {
011186   Mac Pascal heap management
011187  }
011188
011189      PROCEDURE PLHeapInit(sizepheap: LONGINT; heapDelta: LONGINT;
011190                           memerrProc: UNIV PASCALPOINTER; allowNonCont: BOOLEAN;
011191                           forDispose: BOOLEAN);
011192  {
011193   The following procedure is obsolete, use PLHeapInit
011194  }
011195
011196      PROCEDURE PLInitHeap(sizepheap: LONGINT; memerrProc: UNIV PASCALPOINTER;
011197                           allowNonCont: BOOLEAN; allowDispose: BOOLEAN);
011198
011199      PROCEDURE PLSetNonCont(allowNonCont: BOOLEAN);
011200
011201      PROCEDURE PLSetMErrProc(memerrProc: UNIV PASCALPOINTER);
011202
011203      PROCEDURE PLSetHeapType(forDispose: BOOLEAN);
011204
011205      PROCEDURE PLSetHeapCheck(DoIt: BOOLEAN);
011206
011207  {
011208   File I/O
011209  }
011210
011211  (*
011212   *    PROCEDURE
011213   *      RESET(VAR fvar:  UNIV PASCALFILE; OPT fname: STRING);
011214   *      BUILTIN;
011215   *
011216   *    PROCEDURE
011217   *      REWRITE(VAR fvar: UNIV PASCALFILE; OPT fname: STRING);
011218   *      BUILTIN;
011219   *
011220   *    PROCEDURE
011221   *      OPEN(VAR fvar:  UNIV PASCALFILE; fname: STRING);
011222   *      BUILTIN;
011223   *)
011224
011225      PROCEDURE PLSetVBuf(VAR fvar: TEXT; buffer: UNIV PASCALPOINTER;
011226                          style: INTEGER; bufsize: INTEGER);
011227  (*
011228   *    FUNCTION
011229   *      BLOCKREAD(
011230   *        VAR fvar: FILE;
011231   *        VAR buffer: UNIV PASCALBLOCK;
011232   *        nBlocks: INTEGER;
011233   *        OPT stBlock:INTEGER
011234   *      ):
011235   *      INTEGER;
011236   *      BUILTIN;
011237   *
011238   *    FUNCTION
011239   *      BLOCKWRITE(
011240   *        VAR fvar: FILE;
```

```
011241  *         VAR buffer: UNIV PASCALBLOCK;
011242  *         nBlocks: INTEGER;
011243  *         OPT stBlock:INTEGER
011244  *       ):
011245  *       INTEGER;
011246  *       BUILTIN;
011247  *
011248  *     FUNCTION
011249  *       BYTEREAD(
011250  *         VAR fvar: FILE;
011251  *         VAR buffer: UNIV PASCALBLOCK;
011252  *         nBytes:  LONGINT;
011253  *         OPT stByte: LONGINT
011254  *       ):
011255  *       LONGINT;
011256  *       BUILTIN;
011257  *
011258  *     FUNCTION
011259  *       BYTEWRITE(
011260  *         VAR fvar: FILE;
011261  *         VAR buffer: UNIV PASCALBLOCK;
011262  *         nBytes:  LONGINT;
011263  *         OPT stByte: LONGINT
011264  *       ):
011265  *       LONGINT;
011266  *       BUILTIN;
011267  *
011268  *     FUNCTION
011269  *       EOF(OPT VAR fvar: UNIV PASCALFILE):
011270  *       BOOLEAN;
011271  *       BUILTIN;
011272  *
011273  *     FUNCTION
011274  *       EOLN(OPT VAR fvar: TEXT):
011275  *       BOOLEAN;
011276  *       BUILTIN;
011277  *
011278  *     PROCEDURE
011279  *       READ(VAR fvar: TEXT; OPT EXPR_LIST);
011280  *       BUILTIN;
011281  *
011282  *     PROCEDURE
011283  *       READLN(OPT VAR fvar: TEXT; OPT EXPR_LIST);
011284  *       BUILTIN;
011285  *
011286  *     PROCEDURE
011287  *       WRITE(VAR fvar: TEXT; OPT EXPR_LIST);
011288  *       BUILTIN;
011289  *
011290  *     PROCEDURE
011291  *       WRITELN(OPT VAR fvar: TEXT; OPT EXPR_LIST);
011292  *       BUILTIN;
011293  *
011294  *     PROCEDURE
011295  *       GET(VAR fvar: UNIV PASCALFILE);
011296  *       BUILTIN;
```

```
011297   *
011298   *    PROCEDURE
011299   *      PUT(VAR fvar: UNIV PASCALFILE);
011300   *      BUILTIN;
011301   *
011302   *    PROCEDURE
011303   *      SEEK(VAR fvar: UNIV PASCALFILE; recno: LONGINT);
011304   *      BUILTIN;
011305   *)
011306
011307      FUNCTION PLFilePos(VAR fvar: UNIV PASCALFILE): LONGINT;
011308
011309      PROCEDURE PLFlush(VAR fvar: TEXT);
011310
011311      PROCEDURE PLCrunch(VAR fvar: UNIV PASCALFILE);
011312  {
011313   Directory operations.
011314  }
011315
011316      PROCEDURE PLPurge(fname: STRING);
011317
011318      PROCEDURE PLRename(oldFname, newFname: STRING);
011319
011320  {
011321   C string functions for Pascal strings
011322  }
011323      FUNCTION PLStrCmp(string1, string2: STR255): INTEGER;
011324
011325      FUNCTION PLStrnCmp(string1, string2: STR255; n: INTEGER): INTEGER;
011326
011327      FUNCTION PLStrCpy(VAR string1: STR255; string2: STR255): STRINGPTR;
011328
011329      FUNCTION PLStrnCpy(VAR string1: STR255; string2: STR255; n: INTEGER): STRINGPTR;
011330
011331      FUNCTION PLStrCat(VAR string1: STR255; string2: STR255): STRINGPTR;
011332
011333      FUNCTION PLStrnCat(VAR string1: STR255; string2: STR255; n: INTEGER): STRINGPTR;
011334
011335      FUNCTION PLStrChr(string1: STR255; c: CHAR): PTR;
011336
011337      FUNCTION PLStrrChr(string1: STR255; c: CHAR): PTR;
011338
011339      FUNCTION PLStrPBrk(string1, string2: STR255): PTR;
011340
011341      FUNCTION PLStrSpn(string1, string2: STR255): INTEGER;
011342
011343      FUNCTION PLStrStr(string1, string2: STR255): PTR;
011344
011345      FUNCTION PLStrLen(string1: STR255): INTEGER;
011346
011347      FUNCTION PLPos(STRING1: STR255; STRING2: STR255): INTEGER;
011348
011349  {$ENDC}    { UsingPASLIBINTF }
011350
011351  {$IFC NOT UsingIncludes}
011352      END.
```

```
011353  {$ENDC}
011354
011355  ### END OF FILE PasLibIntf.p
011356
```

```
011357
011358  ############################################################################
011359  ### FILE: Perf.p
011360  ############################################################################
011361
011362  {
011363  Created: Monday, January 22, 1990 at 9:18 PM
011364      Perf.p
011365      Pascal Interface to the Macintosh Libraries
011366
011367      Copyright Apple Computer, Inc.  1986-1989
011368      All rights reserved
011369
011370      DESCRIPTION
011371      Provides for PC-sampling of User code resources, ROM code, and RAM (misses).
011372      Produces output text file suitable for input to PerformReport.
011373
011374      Design objectives:
011375      Language independent, i.e. works with Pascal, C, and Assembly.
011376      Covers user resources as well as ROM code.
011377      Memory model independent, i.e. works for Desk Accessories and drivers.
011378      Uses TimeManager on new ROMs, Vertical Blanking interrupt on 64 K ROMs.
011379
011380  }
011381
011382
011383  {$IFC UNDEFINED UsingIncludes}
011384  {$SETC UsingIncludes := 0}
011385  {$ENDC}
011386
011387  {$IFC NOT UsingIncludes}
011388      UNIT Perf;
011389      INTERFACE
011390  {$ENDC}
011391
011392  {$IFC UNDEFINED UsingPerf}
011393  {$SETC UsingPerf := 1}
011394
011395  {$I+}
011396  {$SETC PerfIncludes := UsingIncludes}
011397  {$SETC UsingIncludes := 1}
011398  {$IFC UNDEFINED UsingTypes}
011399  {$I $$Shell(PInterfaces)Types.p}
011400  {$ENDC}
011401  {$SETC UsingIncludes := PerfIncludes}
011402
011403  TYPE
011404
011405  PLongs = ^ALongs;
011406  ALongs = ARRAY [1..8000] OF LONGINT;
011407
011408  PInts = ^AInts;
011409  HInts = ^PInts;
011410
011411  AInts = ARRAY [1..8000] OF INTEGER;
011412
```

```
011413  { PerfGlobals are declared as a record, so main program can allocate
011414  as globals, desk accessory can add to globals allocated via pointer,
011415  print driver can allocate via low memory, etc. }
011416
011417
011418  TP2PerfGlobals = ^TPerfGlobals;
011419  TPerfGlobals = RECORD
011420      startROM: LONGINT;          {ROM Base}
011421      romHits: LONGINT;           {used if MeasureROM is false}
011422      misses: LONGINT;            {count of PC values outside measured memory}
011423      segArray: PLongs;           {array of segment handles}
011424      sizeArray: PLongs;          {array of segment sizes}
011425      idArray: HInts;             {array of segment rsrc IDs}
011426      baseArray: PLongs;          {array of offsets to counters for each segment}
011427      samples: PLongs;            {samples buffer}
011428      buffSize: LONGINT;          {size of samples buffer in bytes}
011429      timeInterval: INTEGER;      {number of clock intervals between interrupts}
011430      bucketSize: INTEGER;        {size of buckets power of 2}
011431      log2buckSize: INTEGER;      {used in CvtPC}
011432      pcOffset: INTEGER;          {offset to the user PC at interrupt time.}
011433      numMeasure: INTEGER;        {# Code segments (w/o jump table)- ROM etc.}
011434      firstCode: INTEGER;         {index of first Code segment}
011435      takingSamples: BOOLEAN;     {true if sampling is enabled}
011436      measureROM: BOOLEAN;
011437      measureCode: BOOLEAN;
011438      ramSeg: INTEGER;            {index of "segment" record to cover RAM > 0 if RAM (misses) are to be bucketed.}
011439      ramBase: LONGINT;           {beginning of RAM being measured.}
011440      measureRAMbucketSize: INTEGER;
011441      measureRAMlog2buckSize: INTEGER;
011442      romVersion: INTEGER;
011443      vRefNum: INTEGER;           {Volume where the report file is to be created}
011444      volumeSelected: BOOLEAN;    {True if user selects the report file name}
011445      rptFileName: Str255;        {Report file name}
011446      rptFileCreator: Str255;     {Report File Creator}
011447      rptFileType: Str255;        {Report File type}
011448      getResType: ResType;        {Resource type}
011449      END;
011450
011451
011452
011453  FUNCTION InitPerf(VAR thePerfGlobals: TP2PerfGlobals;timerCount: INTEGER;
011454      codeAndROMBucketSize: INTEGER;doROM: BOOLEAN;doAppCode: BOOLEAN;appCodeType: Str255;
011455      romID: INTEGER;romName: Str255;doRAM: BOOLEAN;ramLow: LONGINT;ramHigh: LONGINT;
011456      ramBucketSize: INTEGER): BOOLEAN;
011457  { called once to setup Performance monitoring
011458   }
011459
011460  PROCEDURE TermPerf(thePerfGlobals: TP2PerfGlobals);
011461  { if InitPerf succeeds then TermPerf must be called before terminating program.
011462   }
011463
011464  FUNCTION PerfControl(thePerfGlobals: TP2PerfGlobals;turnOn: BOOLEAN): BOOLEAN;
011465  { Call this to turn off/on measuring.
011466   Returns previous state.
011467   }
011468
```

```
011469  FUNCTION PerfDump(thePerfGlobals: TP2PerfGlobals;reportFile: Str255;doHistogram: BOOLEAN;
011470      rptFileColumns: INTEGER): INTEGER;
011471  { Call this to dump the statistics into a file. }
011472
011473
011474  {$ENDC}    { UsingPerf }
011475
011476  {$IFC NOT UsingIncludes}
011477      END.
011478  {$ENDC}
011479
011480
011481  ### END OF FILE Perf.p
011482
```

```
011483
011484  ************************************************************************
011485  *** FILE: Picker.p
011486  ************************************************************************
011487
011488
011489  {
011490  Created: Monday, September 16, 1991 at 12:02 AM
011491   Picker.p
011492   Pascal Interface to the Macintosh Libraries
011493
011494    Copyright Apple Computer, Inc. 1987-1991
011495    All rights reserved
011496  }
011497
011498
011499  {$IFC UNDEFINED UsingIncludes}
011500  {$SETC UsingIncludes := 0}
011501  {$ENDC}
011502
011503  {$IFC NOT UsingIncludes}
011504   UNIT Picker;
011505   INTERFACE
011506  {$ENDC}
011507
011508  {$IFC UNDEFINED UsingPicker}
011509  {$SETC UsingPicker := 1}
011510
011511  {$I+}
011512  {$SETC PickerIncludes := UsingIncludes}
011513  {$SETC UsingIncludes := 1}
011514  {$IFC UNDEFINED UsingQuickdraw}
011515  {$I $$Shell(PInterfaces)Quickdraw.p}
011516  {$ENDC}
011517  {$SETC UsingIncludes := PickerIncludes}
011518
011519  CONST
011520  MaxSmallFract = $0000FFFF;  {Maximum small fract value, as long}
011521
011522  TYPE
011523  { A SmallFract value is just the fractional part of a Fixed number,
011524  which is the low order word.  SmallFracts are used to save room,
011525  and to be compatible with Quickdraw's RGBColor.  They can be
011526  assigned directly to and from INTEGERs. }
011527
011528  SmallFract = INTEGER;       { Unsigned fraction between 0 and 1 }
011529
011530  { For developmental simplicity in switching between the HLS and HSV
011531  models, HLS is reordered into HSL. Thus both models start with
011532  hue and saturation values; value/lightness/brightness is last. }
011533
011534  HSVColor = RECORD
011535   hue: SmallFract;           {Fraction of circle, red at 0}
011536   saturation: SmallFract;    {0-1, 0 for gray, 1 for pure color}
011537   value: SmallFract;         {0-1, 0 for black, 1 for max intensity}
011538   END;
```

```
011539
011540  HSLColor = RECORD
011541   hue: SmallFract;           {Fraction of circle, red at 0}
011542   saturation: SmallFract;    {0-1, 0 for gray, 1 for pure color}
011543   lightness: SmallFract;     {0-1, 0 for black, 1 for white}
011544   END;
011545
011546  CMYColor = RECORD
011547   cyan: SmallFract;
011548   magenta: SmallFract;
011549   yellow: SmallFract;
011550   END;
011551
011552
011553  FUNCTION Fix2SmallFract(f: Fixed): SmallFract;
011554   INLINE $3F3C,$0001,$A82E;
011555  FUNCTION SmallFract2Fix(s: SmallFract): Fixed;
011556   INLINE $3F3C,$0002,$A82E;
011557  PROCEDURE CMY2RGB(cColor: CMYColor;VAR rColor: RGBColor);
011558   INLINE $3F3C,$0003,$A82E;
011559  PROCEDURE RGB2CMY(rColor: RGBColor;VAR cColor: CMYColor);
011560   INLINE $3F3C,$0004,$A82E;
011561  PROCEDURE HSL2RGB(hColor: HSLColor;VAR rColor: RGBColor);
011562   INLINE $3F3C,$0005,$A82E;
011563  PROCEDURE RGB2HSL(rColor: RGBColor;VAR hColor: HSLColor);
011564   INLINE $3F3C,$0006,$A82E;
011565  PROCEDURE HSV2RGB(hColor: HSVColor;VAR rColor: RGBColor);
011566   INLINE $3F3C,$0007,$A82E;
011567  PROCEDURE RGB2HSV(rColor: RGBColor;VAR hColor: HSVColor);
011568   INLINE $3F3C,$0008,$A82E;
011569  FUNCTION GetColor(where: Point;prompt: Str255;inColor: RGBColor;VAR outColor: RGBColor): BOOLEAN;
011570   INLINE $3F3C,$0009,$A82E;
011571
011572
011573  {$ENDC} { UsingPicker }
011574
011575  {$IFC NOT UsingIncludes}
011576   END.
011577  {$ENDC}
011578
011579
011580  ### END OF FILE Picker.p
011581
```

```
011582
011583  ########################################################################
011584  ### FILE: PickerIntf.p
011585  ########################################################################
011586
011587  {
011588      File: PickerIntf.p
011589
011590      As of MPW 3.0, interface files were reorganized to more closely
011591      match "Inside Macintosh" reference books and be more consistant
011592      from language to language.
011593
011594      Interfaces for the Color Picker are now found in Picker.p.
011595      This file, which includes Picker.p, is provided for compatibility
011596      with old sources.
011597
011598      Pascal Interface to the Macintosh Libraries
011599      Copyright Apple Computer, Inc. 1988
011600      All Rights Reserved
011601  }
011602
011603  {$IFC UNDEFINED UsingIncludes}
011604  {$SETC UsingIncludes := 0}
011605  {$ENDC}
011606
011607  {$IFC NOT UsingIncludes}
011608      UNIT PickerIntf;
011609      INTERFACE
011610  {$ENDC}
011611
011612  {$IFC UNDEFINED UsingPickerIntf}
011613  {$SETC UsingPickerIntf := 1}
011614
011615  {$I+}
011616  {$SETC PickerIntfIncludes := UsingIncludes}
011617  {$SETC UsingIncludes := 1}
011618  {$IFC UNDEFINED UsingPicker}
011619  {$I $$Shell(PInterfaces)Picker.p}
011620  {$ENDC}
011621  {$SETC UsingIncludes := PickerIntfIncludes}
011622
011623  {$ENDC}    { UsingPickerIntf }
011624
011625  {$IFC NOT UsingIncludes}
011626      END.
011627  {$ENDC}
011628
011629
011630  ### END OF FILE PickerIntf.p
011631
```

```
011632
011633  ############################################################################
011634  ### FILE: PictUtil.p
011635  ############################################################################
011636
011637
011638  {
011639  Created: Monday, September 16, 1991 at 12:03 AM
011640   PictUtil.p
011641   Pascal Interface to the Macintosh Libraries
011642
011643    Copyright Apple Computer, Inc. 1990-1991
011644    All rights reserved
011645  }
011646
011647
011648  {$IFC UNDEFINED UsingIncludes}
011649  {$SETC UsingIncludes := 0}
011650  {$ENDC}
011651
011652  {$IFC NOT UsingIncludes}
011653   UNIT PictUtil;
011654   INTERFACE
011655  {$ENDC}
011656
011657  {$IFC UNDEFINED UsingPictUtil}
011658  {$SETC UsingPictUtil := 1}
011659
011660  {$I+}
011661  {$SETC PictUtilIncludes := UsingIncludes}
011662  {$SETC UsingIncludes := 1}
011663  {$IFC UNDEFINED UsingTypes}
011664  {$I $$Shell(PInterfaces)Types.p}
011665  {$ENDC}
011666  {$IFC UNDEFINED UsingPalettes}
011667  {$I $$Shell(PInterfaces)Palettes.p}
011668  {$ENDC}
011669  {$SETC UsingIncludes := PictUtilIncludes}
011670
011671  CONST
011672
011673  { verbs for the GetPictInfo, GetPixMapInfo, and NewPictInfo calls }
011674  returnColorTable = 1;
011675  returnPalette = 2;
011676  recordComments = 4;
011677  recordFontInfo = 8;
011678  suppressBlackAndWhite = 16;
011679
011680  { color pick methods }
011681  systemMethod = 0;                   {system color pick method}
011682  popularMethod = 1;                  {method that chooses the most popular set of colors}
011683  medianMethod = 2;                   {method that chooses a good average mix of colors}
011684
011685  { color bank types }
011686  ColorBankIsCustom = -1;
011687  ColorBankIsExactAnd555 = 0;
```

```
011688  ColorBankIs555 = 1;
011689
011690  TYPE
011691  PictInfoID = LONGINT;
011692
011693  CommentSpecPtr = ^CommentSpec;
011694  CommentSpecHandle = ^CommentSpecPtr;
011695  CommentSpec = RECORD
011696   count: INTEGER;                    { number of occurrances of this comment ID }
011697   ID: INTEGER;                       { ID for the comment in the picture }
011698   END;
011699
011700  FontSpecPtr = ^FontSpec;
011701  FontSpecHandle = ^FontSpecPtr;
011702  FontSpec = RECORD
011703   pictFontID: INTEGER;               { ID of the font in the picture }
011704   sysFontID: INTEGER;                { ID of the same font in the current system file }
011705   size: ARRAY [0..3] OF LONGINT;     { bit array of all the sizes found (1..127) (bit 0 means > 127) }
011706   style: INTEGER;                    { combined style of all occurrances of the font }
011707   nameOffset: LONGINT;               { offset into the fontNamesHdl handle for the font's name }
011708   END;
011709
011710  PictInfoPtr = ^PictInfo;
011711  PictInfoHandle = ^PictInfoPtr;
011712  PictInfo = RECORD
011713   version: INTEGER;                  { this is always zero, for now }
011714   uniqueColors: LONGINT;             { the number of actual colors in the picture(s)/pixmap(s) }
011715   thePalette: PaletteHandle;         { handle to the palette information }
011716   theColorTable: CTabHandle;         { handle to the color table }
011717   hRes: Fixed;                       { maximum horizontal resolution for all the pixmaps }
011718   vRes: Fixed;                       { maximum vertical resolution for all the pixmaps }
011719   depth: INTEGER;                    { maximum depth for all the pixmaps (in the picture) }
011720   sourceRect: Rect;                  { the picture frame rectangle (this contains the entire picture) }
011721   textCount: LONGINT;                { total number of text strings in the picture }
011722   lineCount: LONGINT;                { total number of lines in the picture }
011723   rectCount: LONGINT;                { total number of rectangles in the picture }
011724   rRectCount: LONGINT;               { total number of round rectangles in the picture }
011725   ovalCount: LONGINT;                { total number of ovals in the picture }
011726   arcCount: LONGINT;                 { total number of arcs in the picture }
011727   polyCount: LONGINT;                { total number of polygons in the picture }
011728   regionCount: LONGINT;              { total number of regions in the picture }
011729   bitMapCount: LONGINT;              { total number of bitmaps in the picture }
011730   pixMapCount: LONGINT;              { total number of pixmaps in the picture }
011731   commentCount: LONGINT;             { total number of comments in the picture }
011732   uniqueComments: LONGINT;           { the number of unique comments in the picture }
011733   commentHandle: CommentSpecHandle;  { handle to all the comment information }
011734   uniqueFonts: LONGINT;              { the number of unique fonts in the picture }
011735   fontHandle: FontSpecHandle;        { handle to the FontSpec information }
011736   fontNamesHandle: Handle;           { handle to the font names }
011737   reserved1: LONGINT;
011738   reserved2: LONGINT;
011739   END;
011740
011741
011742  FUNCTION GetPictInfo(thePictHandle: PicHandle;
011743                       VAR thePictInfo: PictInfo;
```

```
011744                      verb: INTEGER;
011745                      colorsRequested: INTEGER;
011746                      colorPickMethod: INTEGER;
011747                      version: INTEGER): OSErr;
011748   INLINE $303C,$0800,$A831;
011749  FUNCTION GetPixMapInfo(thePixMapHandle: PixMapHandle;
011750                        VAR thePictInfo: PictInfo;
011751                        verb: INTEGER;
011752                        colorsRequested: INTEGER;
011753                        colorPickMethod: INTEGER;
011754                        version: INTEGER): OSErr;
011755   INLINE $303C,$0801,$A831;
011756  FUNCTION NewPictInfo(VAR thePictInfoID: PictInfoID;
011757                      verb: INTEGER;
011758                      colorsRequested: INTEGER;
011759                      colorPickMethod: INTEGER;
011760                      version: INTEGER): OSErr;
011761   INLINE $303C,$0602,$A831;
011762  FUNCTION RecordPictInfo(thePictInfoID: PictInfoID;
011763                         thePictHandle: PicHandle): OSErr;
011764   INLINE $303C,$0403,$A831;
011765  FUNCTION RecordPixMapInfo(thePictInfoID: PictInfoID;
011766                           thePixMapHandle: PixMapHandle): OSErr;
011767   INLINE $303C,$0404,$A831;
011768  FUNCTION RetrievePictInfo(thePictInfoID: PictInfoID;
011769                           VAR thePictInfo: PictInfo;
011770                           colorsRequested: INTEGER): OSErr;
011771   INLINE $303C,$0505,$A831;
011772  FUNCTION DisposPictInfo(thePictInfoID: PictInfoID): OSErr;
011773   INLINE $303C,$0206,$A831;
011774
011775
011776  {$ENDC} { UsingPictUtil }
011777
011778  {$IFC NOT UsingIncludes}
011779   END.
011780  {$ENDC}
011781
011782
011783  ### END OF FILE PictUtil.p
011784
```

```
011785
011786  ##########################################################################
011787  ### FILE: Power.p
011788  ##########################################################################
011789
011790  {
011791  Created: Sunday, January 6, 1991 at 10:54 PM
011792      Power.p
011793      Pascal Interface to the Macintosh Libraries
011794
011795          Copyright Apple Computer, Inc.    1989-1990
011796          All rights reserved
011797  }
011798
011799
011800  {$IFC UNDEFINED UsingIncludes}
011801  {$SETC UsingIncludes := 0}
011802  {$ENDC}
011803
011804  {$IFC NOT UsingIncludes}
011805      UNIT Power;
011806      INTERFACE
011807  {$ENDC}
011808
011809  {$IFC UNDEFINED UsingPower}
011810  {$SETC UsingPower := 1}
011811
011812  {$I+}
011813  {$SETC PowerIncludes := UsingIncludes}
011814  {$SETC UsingIncludes := 1}
011815  {$IFC UNDEFINED UsingTypes}
011816  {$I $$Shell(PInterfaces)Types.p}
011817  {$ENDC}
011818  {$SETC UsingIncludes := PowerIncludes}
011819
011820  CONST
011821
011822  { Bit positions for ModemByte }
011823  modemOnBit = 0;
011824  ringWakeUpBit = 2;
011825  modemInstalledBit = 3;
011826  ringDetectBit = 4;
011827  modemOnHookBit = 5;
011828
011829  { masks for ModemByte }
011830  modemOnMask = $1;
011831  ringWakeUpMask = $4;
011832  modemInstalledMask = $8;
011833  ringDetectMask = $10;
011834  modemOnHookMask = $20;
011835
011836  { bit positions for BatteryByte }
011837  chargerConnBit = 0;
011838  hiChargeBit = 1;
011839  chargeOverFlowBit = 2;
011840  batteryDeadBit = 3;
```

```
011841  batteryLowBit = 4;
011842  connChangedBit = 5;
011843
011844  { masks for BatteryByte }
011845  chargerConnMask = $1;
011846  hiChargeMask = $2;
011847  chargeOverFlowMask = $4;
011848  batteryDeadMask = $8;
011849  batteryLowMask = $10;
011850  connChangedMask = $20;
011851
011852  { commands to SleepQRec sleepQProc }
011853  sleepRequest = 1;
011854  sleepDemand = 2;
011855  sleepWakeUp = 3;
011856  sleepRevoke = 4;
011857
011858  { SleepQRec.sleepQFlags }
011859  noCalls = 1;
011860  noRequest = 2;
011861
011862  slpQType = 16;
011863  sleepQType = 16;
011864
011865  TYPE
011866  ModemByte = Byte;
011867  BatteryByte = Byte;
011868
011869  PMResultCode = LONGINT;
011870
011871  SleepQRecPtr = ^SleepQRec;
011872  SleepQRec = RECORD
011873      sleepQLink: SleepQRecPtr;
011874      sleepQType: INTEGER;    {type = 16}
011875      sleepQProc: ProcPtr;    {Pointer to sleep routine}
011876      sleepQFlags: INTEGER;
011877      END;
011878
011879
011880  FUNCTION DisableWUTime: OSErr;
011881  FUNCTION GetWUTime(VAR WUTime: LONGINT;VAR WUFlag: Byte): OSErr;
011882  FUNCTION SetWUTime(WUTime: LONGINT): OSErr;
011883  FUNCTION BatteryStatus(VAR Status: Byte;VAR Power: Byte): OSErr;
011884  FUNCTION ModemStatus(VAR Status: Byte): OSErr;
011885  FUNCTION IdleUpdate: LONGINT;
011886      INLINE $A285,$2E80;
011887  FUNCTION GetCPUSpeed: LONGINT;
011888      INLINE $70FF,$A485,$2E80;
011889  PROCEDURE EnableIdle;
011890      INLINE $7000,$A485;
011891  PROCEDURE DisableIdle;
011892      INLINE $7001,$A485;
011893  PROCEDURE SleepQInstall(qRecPtr: SleepQRecPtr);
011894      INLINE $205F,$A28A;
011895  PROCEDURE SleepQRemove(qRecPtr: SleepQRecPtr);
011896      INLINE $205F,$A48A;
```

```
011897  PROCEDURE AOn;
011898      INLINE $7004,$A685;
011899  PROCEDURE AOnIgnoreModem;
011900      INLINE $7005,$A685;
011901  PROCEDURE BOn;
011902      INLINE $7000,$A685;
011903  PROCEDURE AOff;
011904      INLINE $7084,$A685;
011905  PROCEDURE BOff;
011906      INLINE $7080,$A685;
011907
011908
011909  {$ENDC}    { UsingPower }
011910
011911  {$IFC NOT UsingIncludes}
011912      END.
011913  {$ENDC}
011914
011915
011916  ### END OF FILE Power.p
011917
```

```
011918
011919  ############################################################################
011920  ### FILE: PPCToolBox.p
011921  ############################################################################
011922
011923
011924  {
011925  Created: Thursday, September 5, 1991 at 5:57 PM
011926   PPCToolBox.p
011927   Pascal Interface to the Macintosh Libraries
011928
011929    Copyright Apple Computer, Inc. 1989-1991
011930    All rights reserved
011931  }
011932
011933
011934  {$IFC UNDEFINED UsingIncludes}
011935  {$SETC UsingIncludes := 0}
011936  {$ENDC}
011937
011938  {$IFC NOT UsingIncludes}
011939   UNIT PPCToolBox;
011940   INTERFACE
011941  {$ENDC}
011942
011943  {$IFC UNDEFINED UsingPPCToolBox}
011944  {$SETC UsingPPCToolBox := 1}
011945
011946  {$I+}
011947  {$SETC PPCToolBoxIncludes := UsingIncludes}
011948  {$SETC UsingIncludes := 1}
011949  {$IFC UNDEFINED UsingAppleTalk}
011950  {$I $$Shell(PInterfaces)AppleTalk.p}
011951  {$ENDC}
011952  {$IFC UNDEFINED UsingMemory}
011953  {$I $$Shell(PInterfaces)Memory.p}
011954  {$ENDC}
011955  {$IFC UNDEFINED UsingTypes}
011956  {$I $$Shell(PInterfaces)Types.p}
011957  {$ENDC}
011958  {$SETC UsingIncludes := PPCToolBoxIncludes}
011959
011960  CONST
011961
011962  {The following is temporarily placed here, later it will be moved to GestaltEqu}
011963  gestaltPPCSupportsStoreAndForward = $2000;
011964  gestaltPPCVersionAttr = 'ppcv';
011965
011966  TYPE
011967  PPCServiceType = SignedByte;
011968
011969  CONST
011970  ppcServiceRealTime = 1;
011971  ppcServiceStoreAndForward = 2;
011972
011973  TYPE
```

```
011974  PPCLocationKind = INTEGER;
011975
011976  CONST
011977  ppcNoLocation = 0;                          { There is no PPCLocName }
011978  ppcNBPLocation = 1;                         { Use AppleTalk NBP      }
011979  ppcNBPTypeLocation = 2;                     { Used for specifying a location name type during PPCOpen only }
011980
011981  TYPE
011982  PPCPortKinds = INTEGER;
011983
011984  CONST
011985  ppcByCreatorAndType = 1;                    { Port type is specified as colloquial Mac creator and type }
011986  ppcByString = 2;                            { Port type is in pascal string format }
011987
011988  TYPE
011989  PPCSessionOrigin = SignedByte;              { Values returned for request field in PPCInform call }
011990
011991  CONST
011992
011993  { Values returned for requestType field in PPCInform call }
011994  ppcLocalOrigin = 1;                         { session originated from this machine }
011995  ppcRemoteOrigin = 2;                        { session originated from remote machine }
011996
011997  TYPE
011998  PPCPortRefNum = INTEGER;
011999  PPCSessRefNum = LONGINT;
012000
012001  PPCPortPtr = ^PPCPortRec;
012002  PPCPortRec = RECORD
012003   nameScript: ScriptCode;                    { script of name }
012004   name: Str32;                               { name of port as seen in browser }
012005   portKindSelector: PPCPortKinds;            { which variant }
012006   CASE PPCPortKinds OF
012007     ppcByString:
012008    (portTypeStr: Str32);
012009     ppcByCreatorAndType:
012010    (portCreator: OSType;
012011    portType: OSType);
012012   END;
012013
012014  LocationNamePtr = ^LocationNameRec;
012015  LocationNameRec = RECORD
012016   locationKindSelector: PPCLocationKind;     { which variant }
012017   CASE PPCLocationKind OF
012018     ppcNBPLocation:
012019    (nbpEntity: EntityName);                  { NBP name entity }
012020     ppcNBPTypeLocation:
012021    (nbpType: Str32);                         { just the NBP type string, for PPCOpen }
012022   END;
012023
012024  PortInfoPtr = ^PortInfoRec;
012025  PortInfoRec = RECORD
012026   filler1: SignedByte;
012027   authRequired: BOOLEAN;
012028   name: PPCPortRec;
012029   END;
```

```
012030
012031
012032  PortInfoArrayPtr = ^PortInfoArray;
012033  PortInfoArray = ARRAY [0..0] OF PortInfoRec;
012034  { Procedures you will need to write }
012035  PPCFilterProcPtr = ProcPtr;                 { FUNCTION MyPortFilter(locationName: LocationNameRec; thePortInfo: PortInfoRec): BOOLEAN; }
012036  PPCCompProcPtr = ProcPtr;                   { PROCEDURE MyCompletionRoutine(pb: PPCParamBlockPtr); }
012037
012038  PPCOpenPBPtr = ^PPCOpenPBRec;
012039  PPCOpenPBRec = RECORD
012040      qLink:   Ptr;                           { reserved }
012041      csCode:   INTEGER;                      { reserved }
012042      intUse:   INTEGER;                      { reserved }
012043      intUsePtr:  Ptr;                        { reserved }
012044      ioCompletion: PPCCompProcPtr;
012045      ioResult:  OSErr;
012046      reserved:  ARRAY [1..5] OF LONGINT;     { reserved }
012047      portRefNum:  PPCPortRefNum;             { 38 <--   Port Reference          }
012048      filler1:  LONGINT;
012049      serviceType: PPCServiceType;            { 44 -->   Bit field describing the requested port service  }
012050      resFlag:  SignedByte;                   { 45 -->   Must be set to 0          }
012051      portName:  PPCPortPtr;                  { 46 -->   PortName for PPC           }
012052      locationName: LocationNamePtr;          { 50 -->   If NBP  Registration is required       }
012053      networkVisible: BOOLEAN;                { 54 -->   make this network visible on network      }
012054      nbpRegistered: BOOLEAN;                 { 55 <--   The given location name was registered on the network }
012055     END;
012056
012057  PPCInformPBPtr = ^PPCInformPBRec;
012058  PPCInformPBRec = RECORD
012059   qLink: Ptr;                                { reserved }
012060   csCode: INTEGER;                           { reserved }
012061   intUse: INTEGER;                           { reserved }
012062   intUsePtr: Ptr;                            { reserved }
012063   ioCompletion: PPCCompProcPtr;
012064   ioResult: OSErr;
012065   reserved: ARRAY [1..5] OF LONGINT;         { reserved }
012066   portRefNum: PPCPortRefNum;                 { 38 -->   Port Identifier }
012067   sessRefNum: PPCSessRefNum;                 { 40 <--   Session Reference }
012068   serviceType: PPCServiceType;               { 44 <--   Status Flags for type of session, local, remote }
012069   autoAccept: BOOLEAN;                       { 45 -->   if true session will be accepted automatically }
012070   portName: PPCPortPtr;                      { 46 -->   Buffer for Source PPCPortRec }
012071   locationName: LocationNamePtr;             { 50 -->   Buffer for Source LocationNameRec }
012072   userName: StringPtr;                       { 54 -->   Buffer for Soure user's name trying to link. }
012073   userData: LONGINT;                         { 58 <--   value included in PPCStart's userData }
012074   requestType: PPCSessionOrigin;             { 62 <--   Local or Network }
012075   END;
012076
012077  PPCStartPBPtr = ^PPCStartPBRec;
012078  PPCStartPBRec = RECORD
012079   qLink: Ptr;                                { reserved }
012080   csCode: INTEGER;                           { reserved }
012081   intUse: INTEGER;                           { reserved }
012082   intUsePtr: Ptr;                            { reserved }
012083   ioCompletion: PPCCompProcPtr;
012084   ioResult: OSErr;
012085   reserved: ARRAY [1..5] OF LONGINT;         { reserved }
```

```
012086   portRefNum: PPCPortRefNum;                { 38 -->   Port Identifier }
012087   sessRefNum: PPCSessRefNum;                { 40 <--   Session Reference }
012088   serviceType: PPCServiceType;              { 44 <--   Actual service method (realTime) }
012089   resFlag: SignedByte;                      { 45 -->   Must be set to 0  }
012090   portName: PPCPortPtr;                     { 46 -->   Destination portName }
012091   locationName: LocationNamePtr;            { 50 -->   NBP or NAS style service location name }
012092   rejectInfo: LONGINT;                      { 54 <--   reason for rejecting the session request }
012093   userData: LONGINT;                        { 58 -->   Copied to destination PPCInform parameter block }
012094   userRefNum: LONGINT;                      { 62 -->   userRefNum (obtained during login process)  }
012095   END;
012096
012097  PPCAcceptPBPtr = ^PPCAcceptPBRec;
012098  PPCAcceptPBRec = RECORD
012099   qLink: Ptr;                               { reserved }
012100   csCode: INTEGER;                          { reserved }
012101   intUse: INTEGER;                          { reserved }
012102   intUsePtr: Ptr;                           { reserved }
012103   ioCompletion: PPCCompProcPtr;
012104   ioResult: OSErr;
012105   reserved: ARRAY [1..5] OF LONGINT;        { reserved }
012106   filler1: INTEGER;
012107   sessRefNum: PPCSessRefNum;                { 40 -->   Session Reference }
012108   END;
012109
012110  PPCRejectPBPtr = ^PPCRejectPBRec;
012111  PPCRejectPBRec = RECORD
012112   qLink: Ptr;                               { reserved }
012113   csCode: INTEGER;                          { reserved }
012114   intUse: INTEGER;                          { reserved }
012115   intUsePtr: Ptr;                           { reserved }
012116   ioCompletion: PPCCompProcPtr;
012117   ioResult: OSErr;
012118   reserved: ARRAY [1..5] OF LONGINT;        { reserved }
012119   filler1: INTEGER;
012120   sessRefNum: PPCSessRefNum;                { 40 -->   Session Reference }
012121   filler2: INTEGER;
012122   filler3: LONGINT;
012123   filler4: LONGINT;
012124   rejectInfo: LONGINT;                      { 54 -->   reason for rejecting the session request  }
012125   END;
012126
012127  PPCWritePBPtr = ^PPCWritePBRec;
012128  PPCWritePBRec = RECORD
012129   qLink: Ptr;                               { reserved }
012130   csCode: INTEGER;                          { reserved }
012131   intUse: INTEGER;                          { reserved }
012132   intUsePtr: Ptr;                           { reserved }
012133   ioCompletion: PPCCompProcPtr;
012134   ioResult: OSErr;
012135   reserved: ARRAY [1..5] OF LONGINT;        { reserved }
012136   filler1: INTEGER;
012137   sessRefNum: PPCSessRefNum;                { 40 -->   Session Reference }
012138   bufferLength: Size;                       { 44 -->   Length of the message buffer }
012139   actualLength: Size;                       { 48 <--   Actual Length Written }
012140   bufferPtr: Ptr;                           { 52 -->   Pointer to message buffer }
012141   more: BOOLEAN;                            { 56 -->   if more data in this block will be written }
```

```
012142   filler2: SignedByte;
012143   userData: LONGINT;                          { 58 -->   Message block userData Uninterpreted by PPC }
012144   blockCreator: OSType;                       { 62 -->   Message block creator Uninterpreted by PPC }
012145   blockType: OSType;                          { 66 -->   Message block type Uninterpreted by PPC }
012146   END;
012147
012148  PPCReadPBPtr = ^PPCReadPBRec;
012149  PPCReadPBRec = RECORD
012150   qLink: Ptr;                                 { reserved }
012151   csCode: INTEGER;                            { reserved }
012152   intUse: INTEGER;                            { reserved }
012153   intUsePtr: Ptr;                             { reserved }
012154   ioCompletion: PPCCompProcPtr;
012155   ioResult: OSErr;
012156   reserved: ARRAY [1..5] OF LONGINT;          { reserved }
012157   filler1: INTEGER;
012158   sessRefNum: PPCSessRefNum;                  { 40 -->   Session Reference }
012159   bufferLength: Size;                         { 44 -->   Length of the message buffer }
012160   actualLength: Size;                         { 48 <--   Actual length read }
012161   bufferPtr: Ptr;                             { 52 -->   Pointer to message buffer }
012162   more: BOOLEAN;                              { 56 <--   if true more data in this block to be read }
012163   filler2: SignedByte;
012164   userData: LONGINT;                          { 58 <--   Message block userData Uninterpreted by PPC }
012165   blockCreator: OSType;                       { 62 <--   Message block creator Uninterpreted by PPC }
012166   blockType: OSType;                          { 66 <--   Message block type Uninterpreted by PPC }
012167   END;
012168
012169  PPCEndPBPtr = ^PPCEndPBRec;
012170  PPCEndPBRec = RECORD
012171   qLink: Ptr;                                 { reserved }
012172   csCode: INTEGER;                            { reserved }
012173   intUse: INTEGER;                            { reserved }
012174   intUsePtr: Ptr;                             { reserved }
012175   ioCompletion: PPCCompProcPtr;
012176   ioResult: OSErr;
012177   reserved: ARRAY [1..5] OF LONGINT;          { reserved }
012178   filler1: INTEGER;
012179   sessRefNum: PPCSessRefNum;                  { 40 -->   Session Reference }
012180   END;
012181
012182  PPCClosePBPtr = ^PPCClosePBRec;
012183  PPCClosePBRec = RECORD
012184   qLink: Ptr;                                 { reserved }
012185   csCode: INTEGER;                            { reserved }
012186   intUse: INTEGER;                            { reserved }
012187   intUsePtr: Ptr;                             { reserved }
012188   ioCompletion: PPCCompProcPtr;
012189   ioResult: OSErr;
012190   reserved: ARRAY [1..5] OF LONGINT;          { reserved }
012191   portRefNum: PPCPortRefNum;                  { 38 -->   Port Identifier }
012192   END;
012193
012194  IPCListPortsPBPtr = ^IPCListPortsPBRec;
012195  IPCListPortsPBRec = RECORD
012196   qLink: Ptr;                                 { reserved }
012197   csCode: INTEGER;                            { reserved }
```

```
012198   intUse: INTEGER;                          { reserved }
012199   intUsePtr: Ptr;                           { reserved }
012200   ioCompletion: PPCCompProcPtr;
012201   ioResult: OSErr;
012202   reserved: ARRAY [1..5] OF LONGINT;        { reserved }
012203   filler1: INTEGER;
012204   startIndex: INTEGER;                      { 40 -->   Start Index }
012205   requestCount: INTEGER;                    { 42 -->   Number of entries to be returned }
012206   actualCount: INTEGER;                     { 44 <--   Actual Number of entries to be returned }
012207   portName: PPCPortPtr;                     { 46 -->   PortName Match }
012208   locationName: LocationNamePtr;            { 50 -->   NBP or NAS type name to locate the Port Location }
012209   bufferPtr: PortInfoArrayPtr;              { 54 -->   Pointer to a buffer requestCount*sizeof(PortInfo) bytes big }
012210   END;
012211
012212  PPCParamBlockPtr = ^PPCParamBlockRec;
012213  PPCParamBlockRec = RECORD
012214       CASE Integer OF
012215        0: (openParam:   PPCOpenPBRec);
012216        1: (informParam: PPCInformPBRec);
012217        2: (startParam:  PPCStartPBRec);
012218        3: (acceptParam: PPCAcceptPBRec);
012219        4: (rejectParam: PPCRejectPBRec);
012220        5: (writeParam:  PPCWritePBRec);
012221        6: (readParam:   PPCReadPBRec);
012222        7: (endParam:  PPCEndPBRec);
012223        8: (closeParam:  PPCClosePBRec);
012224        9: (listPortsParam: IPCListPortsPBRec);
012225      END;
012226
012227
012228
012229  {  PPC Calling Conventions  }
012230  FUNCTION PPCInit: OSErr;
012231   INLINE $7000,$A0DD,$3E80;
012232  FUNCTION PPCOpen(pb: PPCOpenPBPtr;async: BOOLEAN): OSErr;
012233  FUNCTION PPCOpenSync(pb: PPCOpenPBPtr): OSErr;
012234   INLINE $205F,$7001,$A0DD,$3E80;
012235  FUNCTION PPCOpenAsync(pb: PPCOpenPBPtr): OSErr;
012236   INLINE $205F,$7001,$A4DD,$3E80;
012237  FUNCTION PPCInform(pb: PPCInformPBPtr;async: BOOLEAN): OSErr;
012238  FUNCTION PPCInformSync(pb: PPCInformPBPtr): OSErr;
012239   INLINE $205F,$7003,$A0DD,$3E80;
012240  FUNCTION PPCInformAsync(pb: PPCInformPBPtr): OSErr;
012241   INLINE $205F,$7003,$A4DD,$3E80;
012242  FUNCTION PPCStart(pb: PPCStartPBPtr;async: BOOLEAN): OSErr;
012243  FUNCTION PPCStartSync(pb: PPCStartPBPtr): OSErr;
012244   INLINE $205F,$7002,$A0DD,$3E80;
012245  FUNCTION PPCStartAsync(pb: PPCStartPBPtr): OSErr;
012246   INLINE $205F,$7002,$A4DD,$3E80;
012247  FUNCTION PPCAccept(pb: PPCAcceptPBPtr;async: BOOLEAN): OSErr;
012248  FUNCTION PPCAcceptSync(pb: PPCAcceptPBPtr): OSErr;
012249   INLINE $205F,$7004,$A0DD,$3E80;
012250  FUNCTION PPCAcceptAsync(pb: PPCAcceptPBPtr): OSErr;
012251   INLINE $205F,$7004,$A4DD,$3E80;
012252  FUNCTION PPCReject(pb: PPCRejectPBPtr;async: BOOLEAN): OSErr;
012253  FUNCTION PPCRejectSync(pb: PPCRejectPBPtr): OSErr;
```

```
012254   INLINE $205F,$7005,$A0DD,$3E80;
012255  FUNCTION PPCRejectAsync(pb: PPCRejectPBPtr): OSErr;
012256   INLINE $205F,$7005,$A4DD,$3E80;
012257  FUNCTION PPCWrite(pb: PPCWritePBPtr;async: BOOLEAN): OSErr;
012258  FUNCTION PPCWriteSync(pb: PPCWritePBPtr): OSErr;
012259   INLINE $205F,$7006,$A0DD,$3E80;
012260  FUNCTION PPCWriteAsync(pb: PPCWritePBPtr): OSErr;
012261   INLINE $205F,$7006,$A4DD,$3E80;
012262  FUNCTION PPCRead(pb: PPCReadPBPtr;async: BOOLEAN): OSErr;
012263  FUNCTION PPCReadSync(pb: PPCReadPBPtr): OSErr;
012264   INLINE $205F,$7007,$A0DD,$3E80;
012265  FUNCTION PPCReadAsync(pb: PPCReadPBPtr): OSErr;
012266   INLINE $205F,$7007,$A4DD,$3E80;
012267  FUNCTION PPCEnd(pb: PPCEndPBPtr;async: BOOLEAN): OSErr;
012268  FUNCTION PPCEndSync(pb: PPCEndPBPtr): OSErr;
012269   INLINE $205F,$7008,$A0DD,$3E80;
012270  FUNCTION PPCEndAsync(pb: PPCEndPBPtr): OSErr;
012271   INLINE $205F,$7008,$A4DD,$3E80;
012272  FUNCTION PPCClose(pb: PPCClosePBPtr;async: BOOLEAN): OSErr;
012273  FUNCTION PPCCloseSync(pb: PPCClosePBPtr): OSErr;
012274   INLINE $205F,$7009,$A0DD,$3E80;
012275  FUNCTION PPCCloseAsync(pb: PPCClosePBPtr): OSErr;
012276   INLINE $205F,$7009,$A4DD,$3E80;
012277  FUNCTION IPCListPorts(pb: IPCListPortsPBPtr;async: BOOLEAN): OSErr;
012278  FUNCTION IPCListPortsSync(pb: IPCListPortsPBPtr): OSErr;
012279   INLINE $205F,$700A,$A0DD,$3E80;
012280  FUNCTION IPCListPortsAsync(pb: IPCListPortsPBPtr): OSErr;
012281   INLINE $205F,$700A,$A4DD,$3E80;
012282
012283  FUNCTION PPCKill(pb: PPCParamBlockPtr): OSErr;
012284   INLINE $205F,$700B,$A0DD,$3E80;
012285
012286  FUNCTION DeleteUserIdentity(userRef: LONGINT): OSErr;
012287  FUNCTION GetDefaultUser(VAR userRef: LONGINT;
012288                          VAR userName: Str32): OSErr;
012289  FUNCTION StartSecureSession(pb: PPCStartPBPtr;
012290                              VAR userName: Str32;
012291                              useDefault: BOOLEAN;
012292                              allowGuest: BOOLEAN;
012293                              VAR guestSelected: BOOLEAN;
012294                              prompt: Str255): OSErr;
012295  FUNCTION PPCBrowser(prompt: Str255;
012296                      applListLabel: Str255;
012297                      defaultSpecified: BOOLEAN;
012298                      VAR theLocation: LocationNameRec;
012299                      VAR thePortInfo: PortInfoRec;
012300                      portFilter: PPCFilterProcPtr;
012301                      theLocNBPType: Str32): OSErr;
012302   INLINE $303C,$0D00,$A82B;
012303
012304
012305  {$ENDC} { UsingPPCToolBox }
012306
012307  {$IFC NOT UsingIncludes}
012308   END.
012309  {$ENDC}
```

```
012310
012311
012312  ### END OF FILE PPCToolBox.p
012313
```

```
012314
012315  ############################################################################
012316  ### FILE: Printing.p
012317  ############################################################################
012318
012319
012320  {
012321  Created: Saturday, February 15, 1992 at 11:19 AM
012322   Printing.p
012323   Pascal Interface to the Macintosh Libraries
012324
012325    Copyright Apple Computer, Inc. 1985-1991
012326    All rights reserved
012327  }
012328
012329
012330  {$IFC UNDEFINED UsingIncludes}
012331  {$SETC UsingIncludes := 0}
012332  {$ENDC}
012333
012334  {$IFC NOT UsingIncludes}
012335   UNIT Printing;
012336   INTERFACE
012337  {$ENDC}
012338
012339  {$IFC UNDEFINED UsingPrinting}
012340  {$SETC UsingPrinting := 1}
012341
012342  {$I+}
012343  {$SETC PrintingIncludes := UsingIncludes}
012344  {$SETC UsingIncludes := 1}
012345  {$IFC UNDEFINED UsingQuickdraw}
012346  {$I $$Shell(PInterfaces)Quickdraw.p}
012347  {$ENDC}
012348  {$IFC UNDEFINED UsingDialogs}
012349  {$I $$Shell(PInterfaces)Dialogs.p}
012350  {$ENDC}
012351  {$SETC UsingIncludes := PrintingIncludes}
012352
012353  CONST
012354  iPFMaxPgs = 128;
012355  iPrPgFract = 120;                       {Page scale factor. ptPgSize (below) is in units of 1/iPrPgFract}
012356  iPrPgFst = 1;                           {Page range constants}
012357  iPrPgMax = 9999;
012358  iPrRelease = 3;                         {Current version number of the code.}
012359  iPrSavPFil = -1;
012360  iPrAbort = $0080;
012361  iPrDevCtl = 7;                          {The PrDevCtl Proc's ctl number}
012362  lPrReset = $00010000;                   {The PrDevCtl Proc's CParam for reset}
012363  lPrLineFeed = $00030000;
012364  lPrLFStd = $0003FFFF;                   {The PrDevCtl Proc's CParam for std paper advance}
012365  lPrLFSixth = $0003FFFF;
012366  lPrPageEnd = $00020000;                 {The PrDevCtl Proc's CParam for end page}
012367  lPrDocOpen = $00010000;
012368  lPrPageOpen = $00040000;
012369  lPrPageClose = $00020000;
```

```
012370  lPrDocClose = $00050000;
012371  iFMgrCtl = 8;                           {The FMgr's Tail-hook Proc's ctl number}
012372  iMscCtl = 9;                            {The FMgr's Tail-hook Proc's ctl number}
012373  iPvtCtl = 10;                           {The FMgr's Tail-hook Proc's ctl number}
012374  iMemFullErr = -108;
012375  iIOAbort = -27;
012376  pPrGlobals = $00000944;                 {The PrVars lo mem area:}
012377  bDraftLoop = 0;
012378  bSpoolLoop = 1;
012379  bUser1Loop = 2;
012380  bUser2Loop = 3;
012381  fNewRunBit = 2;
012382  fHiResOK = 3;
012383  fWeOpenedRF = 4;
012384
012385  {Driver constants }
012386  iPrBitsCtl = 4;
012387  lScreenBits = 0;
012388  lPaintBits = 1;
012389  lHiScreenBits = $00000002;              {The Bitmap Print Proc's Screen Bitmap param}
012390  lHiPaintBits = $00000003;               {The Bitmap Print Proc's Paint [sq pix] param}
012391  iPrIOCtl = 5;
012392  iPrEvtCtl = 6;                          {The PrEvent Proc's ctl number}
012393  lPrEvtAll = $0002FFFD;                  {The PrEvent Proc's CParam for the entire screen}
012394  lPrEvtTop = $0001FFFD;                  {The PrEvent Proc's CParam for the top folder}
012395  iPrDrvrRef = -3;
012396  getRslDataOp = 4;
012397  setRslOp = 5;
012398  draftBitsOp = 6;
012399  noDraftBitsOp = 7;
012400  getRotnOp = 8;
012401  NoSuchRsl = 1;
012402  OpNotImpl = 2;                          {the driver doesn't support this opcode}
012403  RgType1 = 1;
012404
012405  TYPE
012406  TFeed = (feedCut,feedFanfold,feedMechCut,feedOther);
012407
012408  TScan = (scanTB,scanBT,scanLR,scanRL);
012409
012410
012411  TPRect = ^Rect;                         { A Rect Ptr }
012412  PrIdleProcPtr = ProcPtr;
012413  PItemProcPtr = ProcPtr;
012414
012415  TPPrPort = ^TPrPort;
012416  TPrPort = RECORD
012417   gPort: GrafPort;                       {The Printer's graf port.}
012418   gProcs: QDProcs;                       {..and its procs}
012419   lGParam1: LONGINT;                     {16 bytes for private parameter storage.}
012420   lGParam2: LONGINT;
012421   lGParam3: LONGINT;
012422   lGParam4: LONGINT;
012423   fOurPtr: BOOLEAN;                      {Whether the PrPort allocation was done by us.}
012424   fOurBits: BOOLEAN;                     {Whether the BitMap allocation was done by us.}
012425   END;
```

```
012426
012427  { Printing Graf Port. All printer imaging, whether spooling, banding, etc, happens "thru" a GrafPort.
012428    This is the "PrPeek" record. }
012429  TPPrInfo = ^TPrInfo;
012430  TPrInfo = RECORD
012431   iDev: INTEGER;                         {Font mgr/QuickDraw device code}
012432   iVRes: INTEGER;                        {Resolution of device, in device coordinates}
012433   iHRes: INTEGER;                        {..note: V before H => compatable with Point.}
012434   rPage: Rect;                           {The page (printable) rectangle in device coordinates.}
012435   END;
012436
012437  { Print Info Record: The parameters needed for page composition. }
012438  TPPrStl = ^TPrStl;
012439  TPrStl = RECORD
012440   wDev: INTEGER;
012441   iPageV: INTEGER;
012442   iPageH: INTEGER;
012443   bPort: SignedByte;
012444   feed: TFeed;
012445   END;
012446
012447  TPPrXInfo = ^TPrXInfo;
012448  TPrXInfo = RECORD
012449   iRowBytes: INTEGER;
012450   iBandV: INTEGER;
012451   iBandH: INTEGER;
012452   iDevBytes: INTEGER;
012453   iBands: INTEGER;
012454   bPatScale: SignedByte;
012455   bUlThick: SignedByte;
012456   bUlOffset: SignedByte;
012457   bUlShadow: SignedByte;
012458   scan: TScan;
012459   bXInfoX: SignedByte;
012460   END;
012461
012462  TPPrJob = ^TPrJob;
012463  TPrJob = RECORD
012464   iFstPage: INTEGER;                     {Page Range.}
012465   iLstPage: INTEGER;
012466   iCopies: INTEGER;                      {No. copies.}
012467   bJDocLoop: SignedByte;                 {The Doc style: Draft, Spool, .., and ..}
012468   fFromUsr: BOOLEAN;                     {Printing from an User's App (not PrApp) flag}
012469   pIdleProc: PrIdleProcPtr;              {The Proc called while waiting on IO etc.}
012470   pFileName: StringPtr;                  {Spool File Name: NIL for default.}
012471   iFileVol: INTEGER;                     {Spool File vol, set to 0 initially}
012472   bFileVers: SignedByte;                 {Spool File version, set to 0 initially}
012473   bJobX: SignedByte;                     {An eXtra byte.}
012474   END;
012475
012476  TPrFlag1 = PACKED RECORD
012477   f15: BOOLEAN;
012478   f14: BOOLEAN;
012479   f13: BOOLEAN;
012480   f12: BOOLEAN;
012481   f11: BOOLEAN;
```

```
012482   f10: BOOLEAN;
012483   f9: BOOLEAN;
012484   f8: BOOLEAN;
012485   f7: BOOLEAN;
012486   f6: BOOLEAN;
012487   f5: BOOLEAN;
012488   f4: BOOLEAN;
012489   f3: BOOLEAN;
012490   f2: BOOLEAN;
012491   fLstPgFst: BOOLEAN;
012492   fUserScale: BOOLEAN;
012493   END;
012494
012495  { Print Job: Print "form" for a single print request. }
012496  TPPrint = ^TPrint;
012497  THPrint = ^TPPrint;
012498  TPrint = RECORD
012499   iPrVersion: INTEGER;                  {(2) Printing software version}
012500   prInfo: TPrInfo;                      {(14) the PrInfo data associated with the current style.}
012501   rPaper: Rect;                         {(8) The paper rectangle [offset from rPage]}
012502   prStl: TPrStl;                        {(8)  This print request's style.}
012503   prInfoPT: TPrInfo;                    {(14)  Print Time Imaging metrics}
012504   prXInfo: TPrXInfo;                    {(16)  Print-time (expanded) Print info record.}
012505   prJob: TPrJob;                        {(20) The Print Job request (82)  Total of the above; 120-82 = 38 bytes needed to fill 120}
012506   CASE INTEGER OF
012507     0:
012508    (printX: ARRAY [1..19] OF INTEGER);
012509     1:
012510    (prFlag1: TPrFlag1;                  {a word of flags}
012511    iZoomMin: INTEGER;
012512    iZoomMax: INTEGER;
012513    hDocName: StringHandle);             {current doc's name, nil = front window}
012514   END;
012515
012516  { The universal 120 byte printing record }
012517  TPPrStatus = ^TPrStatus;
012518  TPrStatus = RECORD
012519   iTotPages: INTEGER;                   {Total pages in Print File.}
012520   iCurPage: INTEGER;                    {Current page number}
012521   iTotCopies: INTEGER;                  {Total copies requested}
012522   iCurCopy: INTEGER;                    {Current copy number}
012523   iTotBands: INTEGER;                   {Total bands per page.}
012524   iCurBand: INTEGER;                    {Current band number}
012525   fPgDirty: BOOLEAN;                    {True if current page has been written to.}
012526   fImaging: BOOLEAN;                    {Set while in band's DrawPic call.}
012527   hPrint: THPrint;                      {Handle to the active Printer record}
012528   pPrPort: TPPrPort;                    {Ptr to the active PrPort}
012529   hPic: PicHandle;                      {Handle to the active Picture}
012530   END;
012531
012532  { Print Status: Print information during printing. }
012533  TPPfPgDir = ^TPfPgDir;
012534  THPfPgDir = ^TPPfPgDir;
012535  TPfPgDir = RECORD
012536   iPages: INTEGER;
012537   iPgPos: ARRAY [0..128] OF LONGINT;     {ARRAY [0..iPfMaxPgs] OF LONGINT}
```

```
012538   END;
012539
012540  { PicFile = a TPfHeader followed by n QuickDraw Pics (whose PicSize is invalid!) }
012541  TPPrDlg = ^TPrDlg;
012542  TPrDlg = RECORD
012543   Dlg: DialogRecord;                    {The Dialog window}
012544   pFltrProc: ModalFilterProcPtr;        {The Filter Proc.}
012545   pItemProc: PItemProcPtr;              {The Item evaluating proc.}
012546   hPrintUsr: THPrint;                   {The user's print record.}
012547   fDoIt: BOOLEAN;
012548   fDone: BOOLEAN;
012549   lUser1: LONGINT;                      {Four longs for user's to hang global data.}
012550   lUser2: LONGINT;                      {...Plus more stuff needed by the particular printing dialog.}
012551   lUser3: LONGINT;
012552   lUser4: LONGINT;
012553   END;
012554
012555
012556  PDlgInitProcPtr = ProcPtr;
012557
012558  TGnlData = RECORD
012559   iOpCode: INTEGER;
012560   iError: INTEGER;
012561   lReserved: LONGINT;                   {more fields here depending on call}
012562   END;
012563
012564  TRslRg = RECORD
012565   iMin: INTEGER;
012566   iMax: INTEGER;
012567   END;
012568
012569  TRslRec = RECORD
012570   iXRsl: INTEGER;
012571   iYRsl: INTEGER;
012572   END;
012573
012574  TGetRslBlk = RECORD
012575   iOpCode: INTEGER;
012576   iError: INTEGER;
012577   lReserved: LONGINT;
012578   iRgType: INTEGER;
012579   xRslRg: TRslRg;
012580   yRslRg: TRslRg;
012581   iRslRecCnt: INTEGER;
012582   rgRslRec: ARRAY [1..27] OF TRslRec;
012583   END;
012584
012585  TSetRslBlk = RECORD
012586   iOpCode: INTEGER;
012587   iError: INTEGER;
012588   lReserved: LONGINT;
012589   hPrint: THPrint;
012590   iXRsl: INTEGER;
012591   iYRsl: INTEGER;
012592   END;
012593
```

```
012594  TDftBitsBlk = RECORD
012595   iOpCode: INTEGER;
012596   iError: INTEGER;
012597   lReserved: LONGINT;
012598   hPrint: THPrint;
012599   END;
012600
012601  TGetRotnBlk = RECORD
012602   iOpCode: INTEGER;
012603   iError: INTEGER;
012604   lReserved: LONGINT;
012605   hPrint: THPrint;
012606   fLandscape: BOOLEAN;
012607   bXtra: SignedByte;
012608   END;
012609
012610  TPBitMap = ^BitMap;                    { A BitMap Ptr }
012611
012612  TN = 0..15;                            { a Nibble }
012613
012614  TPWord = ^TWord;
012615  THWord = ^TPWord;
012616  TWord = PACKED RECORD
012617   CASE INTEGER OF
012618     0:
012619    (c1,c0: CHAR);
012620     1:
012621    (b1,b0: SignedByte);
012622     2:
012623    (usb1,usb0: Byte);
012624     3:
012625    (n3,n2,n1,n0: TN);
012626     4:
012627    (f15,f14,f13,f12,f11,f10,f9,f8,f7,f6,f5,f4,f3,f2,f1,f0: BOOLEAN);
012628     5:
012629    (i0: INTEGER);
012630   END;
012631
012632  TPLong = ^TLong;
012633  THLong = ^TPLong;
012634  TLong = RECORD
012635   CASE INTEGER OF
012636     0:
012637    (w1,w0: TWord);
012638     1:
012639    (b1,b0: LONGINT);
012640     2:
012641    (p0: Ptr);
012642     3:
012643    (h0: Handle);
012644     4:
012645    (pt: Point);
012646   END;
012647
012648
012649  PROCEDURE PrPurge;
```

```
012650   INLINE $2F3C,$A800,$0000,$A8FD;
012651  PROCEDURE PrNoPurge;
012652   INLINE $2F3C,$B000,$0000,$A8FD;
012653  PROCEDURE PrOpen;
012654   INLINE $2F3C,$C800,$0000,$A8FD;
012655  PROCEDURE PrClose;
012656   INLINE $2F3C,$D000,$0000,$A8FD;
012657  PROCEDURE PrintDefault(hPrint: THPrint);
012658   INLINE $2F3C,$2004,$0480,$A8FD;
012659  FUNCTION PrValidate(hPrint: THPrint): BOOLEAN;
012660   INLINE $2F3C,$5204,$0498,$A8FD;
012661  FUNCTION PrStlDialog(hPrint: THPrint): BOOLEAN;
012662   INLINE $2F3C,$2A04,$0484,$A8FD;
012663  FUNCTION PrJobDialog(hPrint: THPrint): BOOLEAN;
012664   INLINE $2F3C,$3204,$0488,$A8FD;
012665  FUNCTION PrStlInit(hPrint: THPrint): TPPrDlg;
012666   INLINE $2F3C,$3C04,$040C,$A8FD;
012667  FUNCTION PrJobInit(hPrint: THPrint): TPPrDlg;
012668   INLINE $2F3C,$4404,$0410,$A8FD;
012669  PROCEDURE PrJobMerge(hPrintSrc: THPrint;hPrintDst: THPrint);
012670   INLINE $2F3C,$5804,$089C,$A8FD;
012671  FUNCTION PrDlgMain(hPrint: THPrint;pDlgInit: PDlgInitProcPtr): BOOLEAN;
012672   INLINE $2F3C,$4A04,$0894,$A8FD;
012673  FUNCTION PrOpenDoc(hPrint: THPrint;pPrPort: TPPrPort;pIOBuf: Ptr): TPPrPort;
012674   INLINE $2F3C,$0400,$0C00,$A8FD;
012675  PROCEDURE PrCloseDoc(pPrPort: TPPrPort);
012676   INLINE $2F3C,$0800,$0484,$A8FD;
012677  PROCEDURE PrOpenPage(pPrPort: TPPrPort;pPageFrame: TPRect);
012678   INLINE $2F3C,$1000,$0808,$A8FD;
012679  PROCEDURE PrClosePage(pPrPort: TPPrPort);
012680   INLINE $2F3C,$1800,$040C,$A8FD;
012681  PROCEDURE PrPicFile(hPrint: THPrint;pPrPort: TPPrPort;pIOBuf: Ptr;pDevBuf: Ptr;
012682   VAR prStatus: TPrStatus);
012683   INLINE $2F3C,$6005,$1480,$A8FD;
012684  FUNCTION PrError: INTEGER;
012685   INLINE $2F3C,$BA00,$0000,$A8FD;
012686  PROCEDURE PrSetError(iErr: INTEGER);
012687   INLINE $2F3C,$C000,$0200,$A8FD;
012688  PROCEDURE PrGeneral(pData: Ptr);
012689   INLINE $2F3C,$7007,$0480,$A8FD;
012690  PROCEDURE PrDrvrOpen;
012691   INLINE $2F3C,$8000,$0000,$A8FD;
012692  PROCEDURE PrDrvrClose;
012693   INLINE $2F3C,$8800,$0000,$A8FD;
012694  PROCEDURE PrCtlCall(iWhichCtl: INTEGER;lParam1: LONGINT;lParam2: LONGINT;
012695   lParam3: LONGINT);
012696   INLINE $2F3C,$A000,$0E00,$A8FD;
012697  FUNCTION PrDrvrDCE: Handle;
012698   INLINE $2F3C,$9400,$0000,$A8FD;
012699  FUNCTION PrDrvrVers: INTEGER;
012700   INLINE $2F3C,$9A00,$0000,$A8FD;
012701
012702
012703  {$ENDC} { UsingPrinting }
012704
012705  {$IFC NOT UsingIncludes}
```

```
012706   END.
012707  {$ENDC}
012708
012709
012710  ### END OF FILE Printing.p
012711
```

```
012712
012713  ############################################################################
012714  ### FILE: PrintTraps.p
012715  ############################################################################
012716
012717  {
012718  Created: Monday, Sempember 9, 1991 at 1:06 PM
012719      PrintTraps.p
012720
012721      Copyright Apple Computer, Inc.  1985-1988
012722      All rights reserved
012723  }
012724
012725  {This file is provided to support existing references to it. The up to date interface is
012726     defined in Printing.p
012727  }
012728
012729  {$IFC UNDEFINED UsingIncludes}
012730  {$SETC UsingIncludes := 0}
012731  {$ENDC}
012732
012733  {$IFC NOT UsingIncludes}
012734      UNIT PrintTraps;
012735      INTERFACE
012736  {$ENDC}
012737
012738  {$IFC UNDEFINED UsingPrintTraps}
012739  {$SETC UsingPrintTraps := 1}
012740
012741  {$I+}
012742  {$SETC PrintTrapsIncludes := UsingIncludes}
012743  {$SETC UsingIncludes := 1}
012744  {$IFC UNDEFINED UsingPrinting}
012745  {$I $$Shell(PInterfaces)Printing.p}
012746  {$ENDC}
012747  {$SETC UsingIncludes := PrintTrapsIncludes}
012748
012749
012750  {$ENDC}    { UsingPrintTraps }
012751
012752  {$IFC NOT UsingIncludes}
012753      END.
012754  {$ENDC}
012755
012756
012757  ### END OF FILE PrintTraps.p
012758
```

```
012759
012760  ##########################################################################
012761  ### FILE: Processes.p
012762  ##########################################################################
012763
012764
012765  {
012766  Created: Monday, September 16, 1991 at 12:12 AM
012767   Processes.p
012768   Pascal Interface to the Macintosh Libraries
012769
012770    Copyright Apple Computer, Inc. 1989-1991
012771    All rights reserved
012772  }
012773
012774
012775  {$IFC UNDEFINED UsingIncludes}
012776  {$SETC UsingIncludes := 0}
012777  {$ENDC}
012778
012779  {$IFC NOT UsingIncludes}
012780   UNIT Processes;
012781   INTERFACE
012782  {$ENDC}
012783
012784  {$IFC UNDEFINED UsingProcesses}
012785  {$SETC UsingProcesses := 1}
012786
012787  {$I+}
012788  {$SETC ProcessesIncludes := UsingIncludes}
012789  {$SETC UsingIncludes := 1}
012790  {$IFC UNDEFINED UsingTypes}
012791  {$I $$Shell(PInterfaces)Types.p}
012792  {$ENDC}
012793  {$IFC UNDEFINED UsingEvents}
012794  {$I $$Shell(PInterfaces)Events.p}
012795  {$ENDC}
012796  {$IFC UNDEFINED UsingFiles}
012797  {$I $$Shell(PInterfaces)Files.p}
012798  {$ENDC}
012799  {$SETC UsingIncludes := ProcessesIncludes}
012800
012801  TYPE
012802  { type for unique process identifier }
012803  ProcessSerialNumberPtr = ^ProcessSerialNumber;
012804  ProcessSerialNumber = RECORD
012805   highLongOfPSN: LONGINT;
012806   lowLongOfPSN: LONGINT;
012807   END;
012808
012809
012810  CONST
012811
012812  {******************************************************************************
012813   *                            Process identifier.
012814   ******************************************************************************
```

```
012815   Various reserved process serial numbers. }
012816
012817  kNoProcess = 0;
012818  kSystemProcess = 1;
012819  kCurrentProcess = 2;
012820
012821  TYPE
012822  {*************************************************************************************************************************************************************
012823   *        Definition of the parameter block passed to _Launch.
012824   ************************************************************************
012825
012826  * Typedef and flags for launchControlFlags field }
012827  LaunchFlags = INTEGER;
012828
012829  CONST
012830
012831  {************************************************************************
012832   *        Definition of the parameter block passed to _Launch.
012833   ************************************************************************}
012834
012835  launchContinue = $4000;
012836  launchNoFileFlags = $0800;
012837  launchUseMinimum = $0400;
012838  launchDontSwitch = $0200;
012839  launchAllow24Bit = $0100;
012840  launchInhibitDaemon = $0080;
012841
012842  TYPE
012843  { Format for first AppleEvent to pass to new process.  The size of the overall
012844   * buffer variable: the message body immediately follows the messageLength.
012845   }
012846  AppParametersPtr = ^AppParameters;
012847  AppParameters = RECORD
012848   theMsgEvent: EventRecord;
012849   eventRefCon: LONGINT;
012850   messageLength: LONGINT;
012851   messageBuffer: ARRAY [0..0] OF SignedByte;
012852   END;
012853
012854  { Parameter block to _Launch }
012855  LaunchPBPtr = ^LaunchParamBlockRec;
012856  LaunchParamBlockRec = RECORD
012857   reserved1: LONGINT;
012858   reserved2: INTEGER;
012859   launchBlockID: INTEGER;
012860   launchEPBLength: LONGINT;
012861   launchFileFlags: INTEGER;
012862   launchControlFlags: LaunchFlags;
012863   launchAppSpec: FSSpecPtr;
012864   launchProcessSN: ProcessSerialNumber;
012865   launchPreferredSize: LONGINT;
012866   launchMinimumSize: LONGINT;
012867   launchAvailableSize: LONGINT;
012868   launchAppParameters: AppParametersPtr;
012869   END;
012870
```

```
012871
012872  CONST
012873
012874  { Set launchBlockID to extendedBlock to specify that extensions exist.
012875  * Set launchEPBLength to extendedBlockLen for compatibility.}
012876
012877
012878
012879
012880  extendedBlock = $4C43;  { 'LC' }
012881  extendedBlockLen = (sizeof(LaunchParamBlockRec) - 12);
012882
012883  {*************************************************************************
012884   * Definition of the information block returned by GetProcessInformation
012885   *************************************************************************
012886   Bits in the processMode field }
012887
012888  modeDeskAccessory = $00020000;
012889  modeMultiLaunch = $00010000;
012890  modeNeedSuspendResume = $00004000;
012891  modeCanBackground = $00001000;
012892  modeDoesActivateOnFGSwitch = $00000800;
012893  modeOnlyBackground = $00000400;
012894  modeGetFrontClicks = $00000200;
012895  modeGetAppDiedMsg = $00000100;
012896  mode32BitCompatible = $00000080;
012897  modeHighLevelEventAware = $00000040;
012898  modeLocalAndRemoteHLEvents = $00000020;
012899  modeStationeryAware = $00000010;
012900  modeUseTextEditServices = $00000008;
012901
012902  TYPE
012903  { Record returned by GetProcessInformation }
012904  ProcessInfoRecPtr = ^ProcessInfoRec;
012905  ProcessInfoRec = RECORD
012906   processInfoLength: LONGINT;
012907   processName: StringPtr;
012908   processNumber: ProcessSerialNumber;
012909   processType: LONGINT;
012910   processSignature: OSType;
012911   processMode: LONGINT;
012912   processLocation: Ptr;
012913   processSize: LONGINT;
012914   processFreeMem: LONGINT;
012915   processLauncher: ProcessSerialNumber;
012916   processLaunchDate: LONGINT;
012917   processActiveTime: LONGINT;
012918   processAppSpec: FSSpecPtr;
012919   END;
012920
012921
012922  FUNCTION LaunchApplication(LaunchParams:LaunchPBPtr):OSErr;
012923   INLINE $205F,$A9F2,$3E80;
012924  FUNCTION LaunchDeskAccessory(pFileSpec: FSSpecPtr;pDAName: StringPtr): OSErr;
012925   INLINE $3F3C,$0036,$A88F;
012926  FUNCTION GetCurrentProcess(VAR PSN: ProcessSerialNumber): OSErr;
```

```
012927   INLINE $3F3C,$0037,$A88F;
012928  FUNCTION GetFrontProcess(VAR PSN: ProcessSerialNumber): OSErr;
012929   INLINE $70FF,$2F00,$3F3C,$0039,$A88F;
012930  FUNCTION GetNextProcess(VAR PSN: ProcessSerialNumber): OSErr;
012931   INLINE $3F3C,$0038,$A88F;
012932  FUNCTION GetProcessInformation(PSN: ProcessSerialNumber;VAR info: ProcessInfoRec): OSErr;
012933   INLINE $3F3C,$003A,$A88F;
012934  FUNCTION SetFrontProcess(PSN: ProcessSerialNumber): OSErr;
012935   INLINE $3F3C,$003B,$A88F;
012936  FUNCTION WakeUpProcess(PSN: ProcessSerialNumber): OSErr;
012937   INLINE $3F3C,$003C,$A88F;
012938  FUNCTION SameProcess(PSN1: ProcessSerialNumber;PSN2: ProcessSerialNumber;
012939   VAR result: BOOLEAN): OSErr;
012940   INLINE $3F3C,$003D,$A88F;
012941
012942
012943  {$ENDC} { UsingProcesses }
012944
012945  {$IFC NOT UsingIncludes}
012946   END.
012947  {$ENDC}
012948
012949
012950  ### END OF FILE Processes.p
012951
```

```
012952
012953  ############################################################################
012954  ### FILE: QDOffscreen.p
012955  ############################################################################
012956
012957
012958  {
012959  Created: Monday, September 16, 1991 at 12:14 AM
012960   QDOffscreen.p
012961   Pascal Interface to the Macintosh Libraries
012962
012963    Copyright Apple Computer, Inc. 1985-1991
012964    All rights reserved
012965  }
012966
012967
012968  {$IFC UNDEFINED UsingIncludes}
012969  {$SETC UsingIncludes := 0}
012970  {$ENDC}
012971
012972  {$IFC NOT UsingIncludes}
012973   UNIT QDOffscreen;
012974   INTERFACE
012975  {$ENDC}
012976
012977  {$IFC UNDEFINED UsingQDOffscreen}
012978  {$SETC UsingQDOffscreen := 1}
012979
012980  {$I+}
012981  {$SETC QDOffscreenIncludes := UsingIncludes}
012982  {$SETC UsingIncludes := 1}
012983  {$IFC UNDEFINED UsingQuickdraw}
012984  {$I $$Shell(PInterfaces)Quickdraw.p}
012985  {$ENDC}
012986  {$SETC UsingIncludes := QDOffscreenIncludes}
012987
012988  CONST
012989
012990  { New error codes }
012991  cDepthErr = -157;   {invalid pixel depth}
012992  pixPurgeBit = 0;
012993  noNewDeviceBit = 1;
012994  useTempMemBit = 2;
012995  keepLocalBit = 3;
012996  pixelsPurgeableBit = 6;
012997  pixelsLockedBit = 7;
012998  mapPixBit = 16;
012999  newDepthBit = 17;
013000  alignPixBit = 18;
013001  newRowBytesBit = 19;
013002  reallocPixBit = 20;
013003  clipPixBit = 28;
013004  stretchPixBit = 29;
013005  ditherPixBit = 30;
013006  gwFlagErrBit = 31;
013007
```

```
013008  TYPE
013009  GWorldFlags = SET OF (pixPurge,noNewDevice,useTempMem,keepLocal,GWorldFlags4,
013010   GWorldFlags5,pixelsPurgeable,pixelsLocked,GWorldFlags8,GWorldFlags9,GWorldFlags10,
013011   GWorldFlags11,GWorldFlags12,GWorldFlags13,GWorldFlags14,GWorldFlags15,
013012   mapPix,newDepth,alignPix,newRowBytes,reallocPix,GWorldFlags21,GWorldFlags22,
013013   GWorldFlags23,GWorldFlags24,GWorldFlags25,GWorldFlags26,GWorldFlags27,
013014   clipPix,stretchPix,ditherPix,gwFlagErr);
013015
013016
013017  { Type definition of a GWorldPtr }
013018  GWorldPtr = CGrafPtr;
013019
013020  FUNCTION NewGWorld(VAR offscreenGWorld: GWorldPtr;PixelDepth: INTEGER;boundsRect: Rect;
013021   cTable: CTabHandle;aGDevice: GDHandle;flags: GWorldFlags): QDErr;
013022   INLINE $203C, $0016, $0000,$AB1D;
013023  FUNCTION LockPixels(pm: PixMapHandle): BOOLEAN;
013024   INLINE $203C, $0004, $0001, $AB1D;
013025  PROCEDURE UnlockPixels(pm: PixMapHandle);
013026   INLINE $203C, $0004, $0002, $AB1D;
013027  FUNCTION UpdateGWorld(VAR offscreenGWorld: GWorldPtr;pixelDepth: INTEGER;
013028   boundsRect: Rect;cTable: CTabHandle;aGDevice: GDHandle;flags: GWorldFlags): GWorldFlags;
013029   INLINE $203C, $0016, $0003, $AB1D;
013030  PROCEDURE DisposeGWorld(offscreenGWorld: GWorldPtr);
013031   INLINE $203C, $0004, $0004, $AB1D;
013032  PROCEDURE GetGWorld(VAR port: CGrafPtr;VAR gdh: GDHandle);
013033   INLINE $203C, $0008, $0005, $AB1D;
013034  PROCEDURE SetGWorld(port: CGrafPtr;gdh: GDHandle);
013035   INLINE $203C, $0008, $0006, $AB1D;
013036  PROCEDURE CTabChanged(ctab: CTabHandle);
013037   INLINE $203C, $0004, $0007, $AB1D;
013038  PROCEDURE PixPatChanged(ppat: PixPatHandle);
013039   INLINE $203C, $0004, $0008, $AB1D;
013040  PROCEDURE PortChanged(port: GrafPtr);
013041   INLINE $203C, $0004, $0009, $AB1D;
013042  PROCEDURE GDeviceChanged(gdh: GDHandle);
013043   INLINE $203C, $0004, $000A, $AB1D;
013044  PROCEDURE AllowPurgePixels(pm: PixMapHandle);
013045   INLINE $203C, $0004, $000B, $AB1D;
013046  PROCEDURE NoPurgePixels(pm: PixMapHandle);
013047   INLINE $203C, $0004, $000C, $AB1D;
013048  FUNCTION GetPixelsState(pm: PixMapHandle): GWorldFlags;
013049   INLINE $203C, $0004, $000D, $AB1D;
013050  PROCEDURE SetPixelsState(pm: PixMapHandle;state: GWorldFlags);
013051   INLINE $203C, $0008, $000E, $AB1D;
013052  FUNCTION GetPixBaseAddr(pm: PixMapHandle): Ptr;
013053   INLINE $203C, $0004, $000F, $AB1D;
013054  FUNCTION NewScreenBuffer(globalRect: Rect;purgeable: BOOLEAN;VAR gdh: GDHandle;
013055   VAR offscreenPixMap: PixMapHandle): QDErr;
013056   INLINE $203C, $000E, $0010, $AB1D;
013057  PROCEDURE DisposeScreenBuffer(offscreenPixMap: PixMapHandle);
013058   INLINE $203C, $0004, $0011, $AB1D;
013059  FUNCTION GetGWorldDevice(offscreenGWorld: GWorldPtr): GDHandle;
013060   INLINE $203C, $0004, $0012, $AB1D;
013061  FUNCTION QDDone(port: GrafPtr): BOOLEAN;
013062   INLINE $203C, $0004, $0013, $AB1D;
013063  FUNCTION OffscreenVersion: LONGINT;
```

```
013064   INLINE $7014, $AB1D;
013065  FUNCTION NewTempScreenBuffer(globalRect: Rect;purgeable: BOOLEAN;VAR gdh: GDHandle;
013066   VAR offscreenPixMap: PixMapHandle): QDErr;
013067   INLINE $203C, $000E, $0015, $AB1D;
013068  FUNCTION PixMap32Bit(pmHandle: PixMapHandle): BOOLEAN;
013069   INLINE $203C, $0004, $0016, $AB1D;
013070  FUNCTION GetGWorldPixMap(offscreenGWorld: GWorldPtr): PixMapHandle;
013071   INLINE $203C, $0004, $0017, $AB1D;
013072
013073
013074  {$ENDC} { UsingQDOffscreen }
013075
013076  {$IFC NOT UsingIncludes}
013077   END.
013078  {$ENDC}
013079
013080
013081  ### END OF FILE QDOffscreen.p
013082
```

```
013083
013084  ############################################################################
013085  ### FILE: Quickdraw.p
013086  ############################################################################
013087
013088  {
013089  Created: Sunday, January 6, 1991 at 10:59 PM
013090      Quickdraw.p
013091      Pascal Interface to the Macintosh Libraries
013092
013093          Copyright Apple Computer, Inc.    1985-1990
013094          All rights reserved
013095  }
013096
013097
013098  {$IFC UNDEFINED UsingIncludes}
013099  {$SETC UsingIncludes := 0}
013100  {$ENDC}
013101
013102  {$IFC NOT UsingIncludes}
013103      UNIT Quickdraw;
013104      INTERFACE
013105  {$ENDC}
013106
013107  {$IFC UNDEFINED UsingQuickdraw}
013108  {$SETC UsingQuickdraw := 1}
013109
013110  {$I+}
013111  {$SETC QuickdrawIncludes := UsingIncludes}
013112  {$SETC UsingIncludes := 1}
013113  {$IFC UNDEFINED UsingTypes}
013114  {$I $$Shell(PInterfaces)Types.p}
013115  {$ENDC}
013116  {$SETC UsingIncludes := QuickdrawIncludes}
013117
013118  CONST
013119  invalColReq = -1;                           {invalid color table request}
013120
013121  { transfer modes }
013122  srcCopy = 0;                                {the 16 transfer modes}
013123  srcOr = 1;
013124  srcXor = 2;
013125  srcBic = 3;
013126  notSrcCopy = 4;
013127  notSrcOr = 5;
013128  notSrcXor = 6;
013129  notSrcBic = 7;
013130  patCopy = 8;
013131  patOr = 9;
013132  patXor = 10;
013133  patBic = 11;
013134  notPatCopy = 12;
013135  notPatOr = 13;
013136  notPatXor = 14;
013137  notPatBic = 15;
013138
```

```
013139  { Special Text Transfer Mode    }
013140  grayishTextOr = 49;
013141
013142  { Arithmetic transfer modes }
013143  blend = 32;
013144  addPin = 33;
013145  addOver = 34;
013146  subPin = 35;
013147  addMax = 37;
013148  adMax = 37;
013149  subOver = 38;
013150  adMin = 39;
013151  ditherCopy = 64;
013152
013153  { Transparent mode constant }
013154  transparent = 36;
013155
013156  { QuickDraw color separation constants }
013157  normalBit = 0;                          {normal screen mapping}
013158  inverseBit = 1;                         {inverse screen mapping}
013159  redBit = 4;                             {RGB additive mapping}
013160  greenBit = 3;
013161  blueBit = 2;
013162  cyanBit = 8;                            {CMYBk subtractive mapping}
013163  magentaBit = 7;
013164  yellowBit = 6;
013165  blackBit = 5;
013166  blackColor = 33;                        {colors expressed in these mappings}
013167  whiteColor = 30;
013168  redColor = 205;
013169  greenColor = 341;
013170  blueColor = 409;
013171  cyanColor = 273;
013172  magentaColor = 137;
013173  yellowColor = 69;
013174
013175  picLParen = 0;                          {standard picture comments}
013176  picRParen = 1;
013177
013178
013179  clutType = 0;                           {0 if lookup table}
013180  fixedType = 1;                          {1 if fixed table}
013181  directType = 2;                         {2 if direct values}
013182
013183  gdDevType = 0;                          {0 = monochrome 1 = color}
013184  burstDevice = 7;
013185  ext32Device = 8;
013186  ramInit = 10;                           {1 if initialized from 'scrn' resource}
013187  mainScreen = 11;                        { 1 if main screen }
013188  allInit = 12;                           { 1 if all devices initialized }
013189  screenDevice = 13;                      {1 if screen device [not used]}
013190  noDriver = 14;                          { 1 if no driver for this GDevice }
013191  screenActive = 15;                      {1 if in use}
013192
013193  hiliteBit = 7;                          {flag bit in HiliteMode (lowMem flag)}
013194  pHiliteBit = 0;                         {flag bit in HiliteMode used with BitClr procedure}
```

```
013195
013196  defQDColors = 127;                          {resource ID of clut for default QDColors}
013197
013198  { pixel type }
013199  RGBDirect = 16;                             { 16 & 32 bits/pixel pixelType value }
013200
013201  { pmVersion values }
013202  baseAddr32 = 4;                             {pixmap base address is 32-bit address}
013203
013204  rgnOverflowErr = -147;                      { Region accumulation failed. Resulting region may be currupt }
013205  insufficientStackErr = -149;                { QuickDraw could not complete the operation }
013206
013207  TYPE
013208  GrafVerb = (frame,paint,erase,invert,fill);
013209
013210  PixelType = (chunky,chunkyPlanar,planar);
013211
013212
013213
013214  PatPtr = ^Pattern;
013215  PatHandle = ^PatPtr;
013216
013217  Pattern = PACKED ARRAY [0..7] OF 0..255;
013218
013219
013220
013221  QDByte = SignedByte;
013222
013223  QDPtr = Ptr;                                { blind pointer }
013224
013225  QDHandle = Handle;                          { blind handle }
013226
013227
013228
013229  QDErr = INTEGER;
013230
013231  Bits16 = ARRAY [0..15] OF INTEGER;
013232
013233
013234  StyleItem = (bold,italic,underline,outline,shadow,condense,extend);
013235
013236
013237  Style = SET OF StyleItem;
013238
013239
013240  DeviceLoopFlags = SET OF (singleDevices,dontMatchSeeds,allDevices,DeviceLoopFlags3,
013241      DeviceLoopFlags4,DeviceLoopFlags5,DeviceLoopFlags6,DeviceLoopFlags7,DeviceLoopFlags8,
013242      DeviceLoopFlags9,DeviceLoopFlags10,DeviceLoopFlags11,DeviceLoopFlags12,
013243      DeviceLoopFlags13,DeviceLoopFlags14,DeviceLoopFlags15,DeviceLoopFlags16,
013244      DeviceLoopFlags17,DeviceLoopFlags18,DeviceLoopFlags19,DeviceLoopFlags20,
013245      DeviceLoopFlags21,DeviceLoopFlags22,DeviceLoopFlags23,DeviceLoopFlags24,
013246      DeviceLoopFlags25,DeviceLoopFlags26,DeviceLoopFlags27,DeviceLoopFlags28,
013247      DeviceLoopFlags29,DeviceLoopFlags30,DeviceLoopFlags31);
013248
013249
013250  FontInfo = RECORD
```

```
013251      ascent: INTEGER;
013252      descent: INTEGER;
013253      widMax: INTEGER;
013254      leading: INTEGER;
013255      END;
013256
013257  BitMapPtr = ^BitMap;
013258  BitMapHandle = ^BitMapPtr;
013259  BitMap = RECORD
013260      baseAddr: Ptr;
013261      rowBytes: INTEGER;
013262      bounds: Rect;
013263      END;
013264
013265  CursPtr = ^Cursor;
013266  CursHandle = ^CursPtr;
013267  Cursor = RECORD
013268      data: Bits16;
013269      mask: Bits16;
013270      hotSpot: Point;
013271      END;
013272
013273  PenState = RECORD
013274      pnLoc: Point;
013275      pnSize: Point;
013276      pnMode: INTEGER;
013277      pnPat: Pattern;
013278      END;
013279
013280  RgnPtr = ^Region;
013281  RgnHandle = ^RgnPtr;
013282  Region = RECORD
013283      rgnSize: INTEGER;                       {size in bytes}
013284      rgnBBox: Rect;                          {enclosing rectangle}
013285      END;
013286
013287  PicPtr = ^Picture;
013288  PicHandle = ^PicPtr;
013289  Picture = RECORD
013290      picSize: INTEGER;
013291      picFrame: Rect;
013292      END;
013293
013294  PolyPtr = ^Polygon;
013295  PolyHandle = ^PolyPtr;
013296  Polygon = RECORD
013297      polySize: INTEGER;
013298      polyBBox: Rect;
013299      polyPoints: ARRAY [0..0] OF Point;
013300      END;
013301
013302  QDProcsPtr = ^QDProcs;
013303  QDProcs = RECORD
013304      textProc: Ptr;
013305      lineProc: Ptr;
013306      rectProc: Ptr;
```

```
013307      rRectProc: Ptr;
013308      ovalProc: Ptr;
013309      arcProc: Ptr;
013310      polyProc: Ptr;
013311      rgnProc: Ptr;
013312      bitsProc: Ptr;
013313      commentProc: Ptr;
013314      txMeasProc: Ptr;
013315      getPicProc: Ptr;
013316      putPicProc: Ptr;
013317      END;
013318
013319  GrafPtr = ^GrafPort;
013320  GrafPort = RECORD
013321      device: INTEGER;
013322      portBits: BitMap;
013323      portRect: Rect;
013324      visRgn: RgnHandle;
013325      clipRgn: RgnHandle;
013326      bkPat: Pattern;
013327      fillPat: Pattern;
013328      pnLoc: Point;
013329      pnSize: Point;
013330      pnMode: INTEGER;
013331      pnPat: Pattern;
013332      pnVis: INTEGER;
013333      txFont: INTEGER;
013334      txFace: Style;                          {txFace is unpacked byte but push as short}
013335      txMode: INTEGER;
013336      txSize: INTEGER;
013337      spExtra: Fixed;
013338      fgColor: LONGINT;
013339      bkColor: LONGINT;
013340      colrBit: INTEGER;
013341      patStretch: INTEGER;
013342      picSave: Handle;
013343      rgnSave: Handle;
013344      polySave: Handle;
013345      grafProcs: QDProcsPtr;
013346      END;
013347
013348
013349  WindowPtr = GrafPtr;
013350
013351  {typedef pascal Boolean (*ColorSearchProcPtr)(RGBColor *rgb, long *position);
013352  typedef pascal Boolean (*ColorComplementProcPtr)(RGBColor *rgb);}
013353
013354  RGBColor = RECORD
013355      red: INTEGER;                           {magnitude of red component}
013356      green: INTEGER;                         {magnitude of green component}
013357      blue: INTEGER;                          {magnitude of blue component}
013358      END;
013359
013360  ColorSpecPtr = ^ColorSpec;
013361  ColorSpec = RECORD
013362      value: INTEGER;                         {index or other value}
```

```
013363      rgb: RGBColor;                          {true color}
013364      END;
013365
013366
013367  CSpecArray = ARRAY [0..0] OF ColorSpec;
013368
013369  CTabPtr = ^ColorTable;
013370  CTabHandle = ^CTabPtr;
013371  ColorTable = RECORD
013372      ctSeed: LONGINT;                        {unique identifier for table}
013373      ctFlags: INTEGER;                       {high bit: 0 = PixMap; 1 = device}
013374      ctSize: INTEGER;                        {number of entries in CTTable}
013375      ctTable: CSpecArray;                    {array [0..0] of ColorSpec}
013376      END;
013377
013378  MatchRec = RECORD
013379      red: INTEGER;
013380      green: INTEGER;
013381      blue: INTEGER;
013382      matchData: LONGINT;
013383      END;
013384
013385  PixMapPtr = ^PixMap;
013386  PixMapHandle = ^PixMapPtr;
013387  PixMap = RECORD
013388      baseAddr: Ptr;                          {pointer to pixels}
013389      rowBytes: INTEGER;                      {offset to next line}
013390      bounds: Rect;                           {encloses bitmap}
013391      pmVersion: INTEGER;                     {pixMap version number}
013392      packType: INTEGER;                      {defines packing format}
013393      packSize: LONGINT;                      {length of pixel data}
013394      hRes: Fixed;                            {horiz. resolution (ppi)}
013395      vRes: Fixed;                            {vert. resolution (ppi)}
013396      pixelType: INTEGER;                     {defines pixel type}
013397      pixelSize: INTEGER;                     {# bits in pixel}
013398      cmpCount: INTEGER;                      {# components in pixel}
013399      cmpSize: INTEGER;                       {# bits per component}
013400      planeBytes: LONGINT;                    {offset to next plane}
013401      pmTable: CTabHandle;                    {color map for this pixMap}
013402      pmReserved: LONGINT;                    {for future use. MUST BE 0}
013403      END;
013404
013405  PixPatPtr = ^PixPat;
013406  PixPatHandle = ^PixPatPtr;
013407  PixPat = RECORD
013408      patType: INTEGER;                       {type of pattern}
013409      patMap: PixMapHandle;                   {the pattern's pixMap}
013410      patData: Handle;                        {pixmap's data}
013411      patXData: Handle;                       {expanded Pattern data}
013412      patXValid: INTEGER;                     {flags whether expanded Pattern valid}
013413      patXMap: Handle;                        {Handle to expanded Pattern data}
013414      pat1Data: Pattern;                      {old-Style pattern/RGB color}
013415      END;
013416
013417  CCrsrPtr = ^CCrsr;
013418  CCrsrHandle = ^CCrsrPtr;
```

```
013419  CCrsr = RECORD
013420      crsrType: INTEGER;                      {type of cursor}
013421      crsrMap: PixMapHandle;                  {the cursor's pixmap}
013422      crsrData: Handle;                       {cursor's data}
013423      crsrXData: Handle;                      {expanded cursor data}
013424      crsrXValid: INTEGER;                    {depth of expanded data (0 if none)}
013425      crsrXHandle: Handle;                    {future use}
013426      crsr1Data: Bits16;                      {one-bit cursor}
013427      crsrMask: Bits16;                       {cursor's mask}
013428      crsrHotSpot: Point;                     {cursor's hotspot}
013429      crsrXTable: LONGINT;                    {private}
013430      crsrID: LONGINT;                        {private}
013431      END;
013432
013433  CIconPtr = ^CIcon;
013434  CIconHandle = ^CIconPtr;
013435  CIcon = RECORD
013436      iconPMap: PixMap;                       {the icon's pixMap}
013437      iconMask: BitMap;                       {the icon's mask}
013438      iconBMap: BitMap;                       {the icon's bitMap}
013439      iconData: Handle;                       {the icon's data}
013440      iconMaskData: ARRAY [0..0] OF INTEGER;  {icon's mask and BitMap data}
013441      END;
013442
013443  GammaTblPtr = ^GammaTbl;
013444  GammaTblHandle = ^GammaTblPtr;
013445  GammaTbl = RECORD
013446      gVersion: INTEGER;                      {gamma version number}
013447      gType: INTEGER;                         {gamma data type}
013448      gFormulaSize: INTEGER;                  {Formula data size}
013449      gChanCnt: INTEGER;                      {number of channels of data}
013450      gDataCnt: INTEGER;                      {number of values/channel}
013451      gDataWidth: INTEGER;                    {bits/corrected value (data packed to next larger byte size)}
013452      gFormulaData: ARRAY [0..0] OF INTEGER;  {data for formulas followed by gamma values}
013453      END;
013454
013455  ITabPtr = ^ITab;
013456  ITabHandle = ^ITabPtr;
013457  ITab = RECORD
013458      iTabSeed: LONGINT;                      {copy of CTSeed from source CTable}
013459      iTabRes: INTEGER;                       {bits/channel resolution of iTable}
013460      iTTable: ARRAY [0..0] OF SignedByte;    {byte colortable index values}
013461      END;
013462
013463  SProcPtr = ^SProcRec;
013464  SProcHndl = ^SProcPtr;
013465  SProcRec = RECORD
013466      nxtSrch: Handle;                        {Handle to next SProcRec}
013467      srchProc: ProcPtr;                      {pointer to search procedure}
013468      END;
013469
013470  CProcPtr = ^CProcRec;
013471  CProcHndl = ^CProcPtr;
013472  CProcRec = RECORD
013473      nxtComp: CProcHndl;                     {CProcHndl Handle to next CProcRec}
013474      compProc: ProcPtr;                      {pointer to complement procedure}
```

```
013475      END;
013476
013477  GDPtr = ^GDevice;
013478  GDHandle = ^GDPtr;
013479  GDevice = RECORD
013480      gdRefNum: INTEGER;                      {driver's unit number}
013481      gdID: INTEGER;                          {client ID for search procs}
013482      gdType: INTEGER;                        {fixed/CLUT/direct}
013483      gdITable: ITabHandle;                   {Handle to inverse lookup table}
013484      gdResPref: INTEGER;                     {preferred resolution of GDITable}
013485      gdSearchProc: SProcHndl;                {search proc list head}
013486      gdCompProc: CProcHndl;                  {complement proc list}
013487      gdFlags: INTEGER;                       {grafDevice flags word}
013488      gdPMap: PixMapHandle;                   {describing pixMap}
013489      gdRefCon: LONGINT;                      {reference value}
013490      gdNextGD: GDHandle;                     {GDHandle Handle of next gDevice}
013491      gdRect: Rect;                           { device's bounds in global coordinates}
013492      gdMode: LONGINT;                        {device's current mode}
013493      gdCCBytes: INTEGER;                     {depth of expanded cursor data}
013494      gdCCDepth: INTEGER;                     {depth of expanded cursor data}
013495      gdCCXData: Handle;                      {Handle to cursor's expanded data}
013496      gdCCXMask: Handle;                      {Handle to cursor's expanded mask}
013497      gdReserved: LONGINT;                    {future use. MUST BE 0}
013498      END;
013499
013500  GVarPtr = ^GrafVars;
013501  GVarHandle = ^GVarPtr;
013502  GrafVars = RECORD
013503      rgbOpColor: RGBColor;                   {color for addPin  subPin and average}
013504      rgbHiliteColor: RGBColor;               {color for hiliting}
013505      pmFgColor: Handle;                      {palette Handle for foreground color}
013506      pmFgIndex: INTEGER;                     {index value for foreground}
013507      pmBkColor: Handle;                      {palette Handle for background color}
013508      pmBkIndex: INTEGER;                     {index value for background}
013509      pmFlags: INTEGER;                       {flags for Palette Manager}
013510      END;
013511
013512  CQDProcsPtr = ^CQDProcs;
013513  CQDProcs = RECORD
013514      textProc: Ptr;
013515      lineProc: Ptr;
013516      rectProc: Ptr;
013517      rRectProc: Ptr;
013518      ovalProc: Ptr;
013519      arcProc: Ptr;
013520      polyProc: Ptr;
013521      rgnProc: Ptr;
013522      bitsProc: Ptr;
013523      commentProc: Ptr;
013524      txMeasProc: Ptr;
013525      getPicProc: Ptr;
013526      putPicProc: Ptr;
013527      opcodeProc: Ptr;                        {fields added to QDProcs}
013528      newProc1: Ptr;
013529      newProc2: Ptr;
013530      newProc3: Ptr;
```

```
013531      newProc4: Ptr;
013532      newProc5: Ptr;
013533      newProc6: Ptr;
013534      END;
013535
013536  CGrafPtr = ^CGrafPort;
013537  CGrafPort = RECORD
013538      device: INTEGER;
013539      portPixMap: PixMapHandle;               {port's pixel map}
013540      portVersion: INTEGER;                   {high 2 bits always set}
013541      grafVars: Handle;                       {Handle to more fields}
013542      chExtra: INTEGER;                       {character extra}
013543      pnLocHFrac: INTEGER;                    {pen fraction}
013544      portRect: Rect;
013545      visRgn: RgnHandle;
013546      clipRgn: RgnHandle;
013547      bkPixPat: PixPatHandle;                 {background pattern}
013548      rgbFgColor: RGBColor;                   {RGB components of fg}
013549      rgbBkColor: RGBColor;                   {RGB components of bk}
013550      pnLoc: Point;
013551      pnSize: Point;
013552      pnMode: INTEGER;
013553      pnPixPat: PixPatHandle;                 {pen's pattern}
013554      fillPixPat: PixPatHandle;               {fill pattern}
013555      pnVis: INTEGER;
013556      txFont: INTEGER;
013557      txFace: Style;                          {txFace is unpacked byte  push as short}
013558      txMode: INTEGER;
013559      txSize: INTEGER;
013560      spExtra: Fixed;
013561      fgColor: LONGINT;
013562      bkColor: LONGINT;
013563      colrBit: INTEGER;
013564      patStretch: INTEGER;
013565      picSave: Handle;
013566      rgnSave: Handle;
013567      polySave: Handle;
013568      grafProcs: CQDProcsPtr;
013569      END;
013570
013571
013572  CWindowPtr = CGrafPtr;
013573
013574  ReqListRec = RECORD
013575      reqLSize: INTEGER;                      {request list size}
013576      reqLData: ARRAY [0..0] OF INTEGER;      {request list data}
013577      END;
013578
013579  OpenCPicParams = RECORD
013580      srcRect: Rect;
013581      hRes: Fixed;
013582      vRes: Fixed;
013583      version: INTEGER;
013584      reserved1: INTEGER;
013585      reserved2: LONGINT;
013586      END;
```

```
013587
013588
013589  DeviceLoopDrawingProcPtr = ProcPtr;
013590
013591
013592  VAR
013593  {$PUSH}
013594  {$J+}
013595      thePort: GrafPtr;
013596      white: Pattern;
013597      black: Pattern;
013598      gray: Pattern;
013599      ltGray: Pattern;
013600      dkGray: Pattern;
013601      arrow: Cursor;
013602      screenBits: BitMap;
013603      randSeed: LONGINT;
013604  {$POP}
013605
013606
013607  PROCEDURE InitGraf(globalPtr: Ptr);
013608      INLINE $A86E;
013609  PROCEDURE OpenPort(port: GrafPtr);
013610      INLINE $A86F;
013611  PROCEDURE InitPort(port: GrafPtr);
013612      INLINE $A86D;
013613  PROCEDURE ClosePort(port: GrafPtr);
013614      INLINE $A87D;
013615  PROCEDURE SetPort(port: GrafPtr);
013616      INLINE $A873;
013617  PROCEDURE GetPort(VAR port: GrafPtr);
013618      INLINE $A874;
013619  PROCEDURE GrafDevice(device: INTEGER);
013620      INLINE $A872;
013621  PROCEDURE SetPortBits(bm: BitMap);
013622      INLINE $A875;
013623  PROCEDURE PortSize(width: INTEGER;height: INTEGER);
013624      INLINE $A876;
013625  PROCEDURE MovePortTo(leftGlobal: INTEGER;topGlobal: INTEGER);
013626      INLINE $A877;
013627  PROCEDURE SetOrigin(h: INTEGER;v: INTEGER);
013628      INLINE $A878;
013629  PROCEDURE SetClip(rgn: RgnHandle);
013630      INLINE $A879;
013631  PROCEDURE GetClip(rgn: RgnHandle);
013632      INLINE $A87A;
013633  PROCEDURE ClipRect(r: Rect);
013634      INLINE $A87B;
013635  PROCEDURE BackPat(pat: Pattern);
013636      INLINE $A87C;
013637  PROCEDURE InitCursor;
013638      INLINE $A850;
013639  PROCEDURE SetCursor(crsr: Cursor);
013640      INLINE $A851;
013641  PROCEDURE HideCursor;
013642      INLINE $A852;
```

```
013643  PROCEDURE ShowCursor;
013644      INLINE $A853;
013645  PROCEDURE ObscureCursor;
013646      INLINE $A856;
013647  PROCEDURE HidePen;
013648      INLINE $A896;
013649  PROCEDURE ShowPen;
013650      INLINE $A897;
013651  PROCEDURE GetPen(VAR pt: Point);
013652      INLINE $A89A;
013653  PROCEDURE GetPenState(VAR pnState: PenState);
013654      INLINE $A898;
013655  PROCEDURE SetPenState(pnState: PenState);
013656      INLINE $A899;
013657  PROCEDURE PenSize(width: INTEGER;height: INTEGER);
013658      INLINE $A89B;
013659  PROCEDURE PenMode(mode: INTEGER);
013660      INLINE $A89C;
013661  PROCEDURE PenPat(pat: Pattern);
013662      INLINE $A89D;
013663  PROCEDURE PenNormal;
013664      INLINE $A89E;
013665  PROCEDURE MoveTo(h: INTEGER;v: INTEGER);
013666      INLINE $A893;
013667  PROCEDURE Move(dh: INTEGER;dv: INTEGER);
013668      INLINE $A894;
013669  PROCEDURE LineTo(h: INTEGER;v: INTEGER);
013670      INLINE $A891;
013671  PROCEDURE Line(dh: INTEGER;dv: INTEGER);
013672      INLINE $A892;
013673  PROCEDURE TextFont(font: INTEGER);
013674      INLINE $A887;
013675  PROCEDURE TextFace(face: Style);
013676      INLINE $A888;
013677  PROCEDURE TextMode(mode: INTEGER);
013678      INLINE $A889;
013679  PROCEDURE TextSize(size: INTEGER);
013680      INLINE $A88A;
013681  PROCEDURE SpaceExtra(extra: Fixed);
013682      INLINE $A88E;
013683  PROCEDURE DrawChar(ch: CHAR);
013684      INLINE $A883;
013685  PROCEDURE DrawString(s: Str255);
013686      INLINE $A884;
013687  PROCEDURE DrawText(textBuf: Ptr;firstByte: INTEGER;byteCount: INTEGER);
013688      INLINE $A885;
013689  FUNCTION CharWidth(ch: CHAR): INTEGER;
013690      INLINE $A88D;
013691  FUNCTION StringWidth(s: Str255): INTEGER;
013692      INLINE $A88C;
013693  FUNCTION TextWidth(textBuf: Ptr;firstByte: INTEGER;byteCount: INTEGER): INTEGER;
013694      INLINE $A886;
013695  PROCEDURE MeasureText(count: INTEGER;textAddr: Ptr;charLocs: Ptr);
013696      INLINE $A837;
013697  PROCEDURE GetFontInfo(VAR info: FontInfo);
013698      INLINE $A88B;
```

```
013699  PROCEDURE ForeColor(color: LONGINT);
013700      INLINE $A862;
013701  PROCEDURE BackColor(color: LONGINT);
013702      INLINE $A863;
013703  PROCEDURE ColorBit(whichBit: INTEGER);
013704      INLINE $A864;
013705  PROCEDURE SetRect(VAR r: Rect;left: INTEGER;top: INTEGER;right: INTEGER;
013706      bottom: INTEGER);
013707      INLINE $A8A7;
013708  PROCEDURE OffsetRect(VAR r: Rect;dh: INTEGER;dv: INTEGER);
013709      INLINE $A8A8;
013710  PROCEDURE InsetRect(VAR r: Rect;dh: INTEGER;dv: INTEGER);
013711      INLINE $A8A9;
013712  FUNCTION SectRect(src1: Rect;src2: Rect;VAR dstRect: Rect): BOOLEAN;
013713      INLINE $A8AA;
013714  PROCEDURE UnionRect(src1: Rect;src2: Rect;VAR dstRect: Rect);
013715      INLINE $A8AB;
013716  FUNCTION EqualRect(rect1: Rect;rect2: Rect): BOOLEAN;
013717      INLINE $A8A6;
013718  FUNCTION EmptyRect(r: Rect): BOOLEAN;
013719      INLINE $A8AE;
013720  PROCEDURE FrameRect(r: Rect);
013721      INLINE $A8A1;
013722  PROCEDURE PaintRect(r: Rect);
013723      INLINE $A8A2;
013724  PROCEDURE EraseRect(r: Rect);
013725      INLINE $A8A3;
013726  PROCEDURE InvertRect(r: Rect);
013727      INLINE $A8A4;
013728  PROCEDURE FillRect(r: Rect;pat: Pattern);
013729      INLINE $A8A5;
013730  PROCEDURE FrameOval(r: Rect);
013731      INLINE $A8B7;
013732  PROCEDURE PaintOval(r: Rect);
013733      INLINE $A8B8;
013734  PROCEDURE EraseOval(r: Rect);
013735      INLINE $A8B9;
013736  PROCEDURE InvertOval(r: Rect);
013737      INLINE $A8BA;
013738  PROCEDURE FillOval(r: Rect;pat: Pattern);
013739      INLINE $A8BB;
013740  PROCEDURE FrameRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER);
013741      INLINE $A8B0;
013742  PROCEDURE PaintRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER);
013743      INLINE $A8B1;
013744  PROCEDURE EraseRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER);
013745      INLINE $A8B2;
013746  PROCEDURE InvertRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER);
013747      INLINE $A8B3;
013748  PROCEDURE FillRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER;
013749      pat: Pattern);
013750      INLINE $A8B4;
013751  PROCEDURE FrameArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER);
013752      INLINE $A8BE;
013753  PROCEDURE PaintArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER);
013754      INLINE $A8BF;
```

```
013755  PROCEDURE EraseArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER);
013756      INLINE $A8C0;
013757  PROCEDURE InvertArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER);
013758      INLINE $A8C1;
013759  PROCEDURE FillArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER;pat: Pattern);
013760      INLINE $A8C2;
013761  FUNCTION NewRgn: RgnHandle;
013762      INLINE $A8D8;
013763  PROCEDURE OpenRgn;
013764      INLINE $A8DA;
013765  PROCEDURE CloseRgn(dstRgn: RgnHandle);
013766      INLINE $A8DB;
013767  FUNCTION BitMapToRegionGlue(region: RgnHandle;bMap: BitMap): OSErr;
013768  FUNCTION BitMapToRegion(region: RgnHandle;bMap: BitMap): OSErr;
013769      INLINE $A8D7;
013770  PROCEDURE DisposeRgn(rgn: RgnHandle);
013771      INLINE $A8D9;
013772  PROCEDURE CopyRgn(srcRgn: RgnHandle;dstRgn: RgnHandle);
013773      INLINE $A8DC;
013774  PROCEDURE SetEmptyRgn(rgn: RgnHandle);
013775      INLINE $A8DD;
013776  PROCEDURE SetRectRgn(rgn: RgnHandle;left: INTEGER;top: INTEGER;right: INTEGER;
013777      bottom: INTEGER);
013778      INLINE $A8DE;
013779  PROCEDURE RectRgn(rgn: RgnHandle;r: Rect);
013780      INLINE $A8DF;
013781  PROCEDURE OffsetRgn(rgn: RgnHandle;dh: INTEGER;dv: INTEGER);
013782      INLINE $A8E0;
013783  PROCEDURE InsetRgn(rgn: RgnHandle;dh: INTEGER;dv: INTEGER);
013784      INLINE $A8E1;
013785  PROCEDURE SectRgn(srcRgnA: RgnHandle;srcRgnB: RgnHandle;dstRgn: RgnHandle);
013786      INLINE $A8E4;
013787  PROCEDURE UnionRgn(srcRgnA: RgnHandle;srcRgnB: RgnHandle;dstRgn: RgnHandle);
013788      INLINE $A8E5;
013789  PROCEDURE DiffRgn(srcRgnA: RgnHandle;srcRgnB: RgnHandle;dstRgn: RgnHandle);
013790      INLINE $A8E6;
013791  PROCEDURE XorRgn(srcRgnA: RgnHandle;srcRgnB: RgnHandle;dstRgn: RgnHandle);
013792      INLINE $A8E7;
013793  FUNCTION RectInRgn(r: Rect;rgn: RgnHandle): BOOLEAN;
013794      INLINE $A8E9;
013795  FUNCTION EqualRgn(rgnA: RgnHandle;rgnB: RgnHandle): BOOLEAN;
013796      INLINE $A8E3;
013797  FUNCTION EmptyRgn(rgn: RgnHandle): BOOLEAN;
013798      INLINE $A8E2;
013799  PROCEDURE FrameRgn(rgn: RgnHandle);
013800      INLINE $A8D2;
013801  PROCEDURE PaintRgn(rgn: RgnHandle);
013802      INLINE $A8D3;
013803  PROCEDURE EraseRgn(rgn: RgnHandle);
013804      INLINE $A8D4;
013805  PROCEDURE InvertRgn(rgn: RgnHandle);
013806      INLINE $A8D5;
013807  PROCEDURE FillRgn(rgn: RgnHandle;pat: Pattern);
013808      INLINE $A8D6;
013809  PROCEDURE ScrollRect(r: Rect;dh: INTEGER;dv: INTEGER;updateRgn: RgnHandle);
013810      INLINE $A8EF;
```

```
013811  PROCEDURE CopyBits(srcBits: BitMap;dstBits: BitMap;srcRect: Rect;dstRect: Rect;
013812      mode: INTEGER;maskRgn: RgnHandle);
013813      INLINE $A8EC;
013814  PROCEDURE SeedFill(srcPtr: Ptr;dstPtr: Ptr;srcRow: INTEGER;dstRow: INTEGER;
013815      height: INTEGER;words: INTEGER;seedH: INTEGER;seedV: INTEGER);
013816      INLINE $A839;
013817  PROCEDURE CalcMask(srcPtr: Ptr;dstPtr: Ptr;srcRow: INTEGER;dstRow: INTEGER;
013818      height: INTEGER;words: INTEGER);
013819      INLINE $A838;
013820  PROCEDURE CopyMask(srcBits: BitMap;maskBits: BitMap;dstBits: BitMap;srcRect: Rect;
013821      maskRect: Rect;dstRect: Rect);
013822      INLINE $A817;
013823  FUNCTION OpenPicture(picFrame: Rect): PicHandle;
013824      INLINE $A8F3;
013825  PROCEDURE PicComment(kind: INTEGER;dataSize: INTEGER;dataHandle: Handle);
013826      INLINE $A8F2;
013827  PROCEDURE ClosePicture;
013828      INLINE $A8F4;
013829  PROCEDURE DrawPicture(myPicture: PicHandle;dstRect: Rect);
013830      INLINE $A8F6;
013831  PROCEDURE KillPicture(myPicture: PicHandle);
013832      INLINE $A8F5;
013833  FUNCTION OpenPoly: PolyHandle;
013834      INLINE $A8CB;
013835  PROCEDURE ClosePoly;
013836      INLINE $A8CC;
013837  PROCEDURE KillPoly(poly: PolyHandle);
013838      INLINE $A8CD;
013839  PROCEDURE OffsetPoly(poly: PolyHandle;dh: INTEGER;dv: INTEGER);
013840      INLINE $A8CE;
013841  PROCEDURE FramePoly(poly: PolyHandle);
013842      INLINE $A8C6;
013843  PROCEDURE PaintPoly(poly: PolyHandle);
013844      INLINE $A8C7;
013845  PROCEDURE ErasePoly(poly: PolyHandle);
013846      INLINE $A8C8;
013847  PROCEDURE InvertPoly(poly: PolyHandle);
013848      INLINE $A8C9;
013849  PROCEDURE FillPoly(poly: PolyHandle;pat: Pattern);
013850      INLINE $A8CA;
013851  PROCEDURE SetPt(VAR pt: Point;h: INTEGER;v: INTEGER);
013852      INLINE $A880;
013853  PROCEDURE LocalToGlobal(VAR pt: Point);
013854      INLINE $A870;
013855  PROCEDURE GlobalToLocal(VAR pt: Point);
013856      INLINE $A871;
013857  FUNCTION Random: INTEGER;
013858      INLINE $A861;
013859  PROCEDURE StuffHex(thingPtr: Ptr;s: Str255);
013860      INLINE $A866;
013861  FUNCTION GetPixel(h: INTEGER;v: INTEGER): BOOLEAN;
013862      INLINE $A865;
013863  PROCEDURE ScalePt(VAR pt: Point;srcRect: Rect;dstRect: Rect);
013864      INLINE $A8F8;
013865  PROCEDURE MapPt(VAR pt: Point;srcRect: Rect;dstRect: Rect);
013866      INLINE $A8F9;
```

```
013867  PROCEDURE MapRect(VAR r: Rect;srcRect: Rect;dstRect: Rect);
013868      INLINE $A8FA;
013869  PROCEDURE MapRgn(rgn: RgnHandle;srcRect: Rect;dstRect: Rect);
013870      INLINE $A8FB;
013871  PROCEDURE MapPoly(poly: PolyHandle;srcRect: Rect;dstRect: Rect);
013872      INLINE $A8FC;
013873  PROCEDURE SetStdProcs(VAR procs: QDProcs);
013874      INLINE $A8EA;
013875  PROCEDURE StdRect(verb: GrafVerb;r: Rect);
013876      INLINE $A8A0;
013877  PROCEDURE StdRRect(verb: GrafVerb;r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER);
013878      INLINE $A8AF;
013879  PROCEDURE StdOval(verb: GrafVerb;r: Rect);
013880      INLINE $A8B6;
013881  PROCEDURE StdArc(verb: GrafVerb;r: Rect;startAngle: INTEGER;arcAngle: INTEGER);
013882      INLINE $A8BD;
013883  PROCEDURE StdPoly(verb: GrafVerb;poly: PolyHandle);
013884      INLINE $A8C5;
013885  PROCEDURE StdRgn(verb: GrafVerb;rgn: RgnHandle);
013886      INLINE $A8D1;
013887  PROCEDURE StdBits(VAR srcBits: BitMap;VAR srcRect: Rect;dstRect: Rect;mode: INTEGER;
013888      maskRgn: RgnHandle);
013889      INLINE $A8EB;
013890  PROCEDURE StdComment(kind: INTEGER;dataSize: INTEGER;dataHandle: Handle);
013891      INLINE $A8F1;
013892  FUNCTION StdTxMeas(byteCount: INTEGER;textAddr: Ptr;VAR numer: Point;VAR denom: Point;
013893      VAR info: FontInfo): INTEGER;
013894      INLINE $A8ED;
013895  PROCEDURE StdGetPic(dataPtr: Ptr;byteCount: INTEGER);
013896      INLINE $A8EE;
013897  PROCEDURE StdPutPic(dataPtr: Ptr;byteCount: INTEGER);
013898      INLINE $A8F0;
013899  PROCEDURE AddPt(src: Point;VAR dst: Point);
013900      INLINE $A87E;
013901  FUNCTION EqualPt(pt1: Point;pt2: Point): BOOLEAN;
013902      INLINE $A881;
013903  FUNCTION PtInRect(pt: Point;r: Rect): BOOLEAN;
013904      INLINE $A8AD;
013905  PROCEDURE Pt2Rect(pt1: Point;pt2: Point;VAR dstRect: Rect);
013906      INLINE $A8AC;
013907  PROCEDURE PtToAngle(r: Rect;pt: Point;VAR angle: INTEGER);
013908      INLINE $A8C3;
013909  FUNCTION PtInRgn(pt: Point;rgn: RgnHandle): BOOLEAN;
013910      INLINE $A8E8;
013911  PROCEDURE StdText(count: INTEGER;textAddr: Ptr;numer: Point;denom: Point);
013912      INLINE $A882;
013913  PROCEDURE StdLine(newPt: Point);
013914      INLINE $A890;
013915  PROCEDURE OpenCPort(port: CGrafPtr);
013916      INLINE $AA00;
013917  PROCEDURE InitCPort(port: CGrafPtr);
013918      INLINE $AA01;
013919  PROCEDURE CloseCPort(port: CGrafPtr);
013920      INLINE $A87D;
013921  FUNCTION NewPixMap: PixMapHandle;
013922      INLINE $AA03;
```

```
013923  PROCEDURE DisposPixMap(pm: PixMapHandle);
013924      INLINE $AA04;
013925  PROCEDURE DisposePixMap(pm: PixMapHandle);
013926      INLINE $AA04;
013927  PROCEDURE CopyPixMap(srcPM: PixMapHandle;dstPM: PixMapHandle);
013928      INLINE $AA05;
013929  FUNCTION NewPixPat: PixPatHandle;
013930      INLINE $AA07;
013931  PROCEDURE DisposPixPat(pp: PixPatHandle);
013932      INLINE $AA08;
013933  PROCEDURE DisposePixPat(pp: PixPatHandle);
013934      INLINE $AA08;
013935  PROCEDURE CopyPixPat(srcPP: PixPatHandle;dstPP: PixPatHandle);
013936      INLINE $AA09;
013937  PROCEDURE PenPixPat(pp: PixPatHandle);
013938      INLINE $AA0A;
013939  PROCEDURE BackPixPat(pp: PixPatHandle);
013940      INLINE $AA0B;
013941  FUNCTION GetPixPat(patID: INTEGER): PixPatHandle;
013942      INLINE $AA0C;
013943  PROCEDURE MakeRGBPat(pp: PixPatHandle;myColor: RGBColor);
013944      INLINE $AA0D;
013945  PROCEDURE FillCRect(r: Rect;pp: PixPatHandle);
013946      INLINE $AA0E;
013947  PROCEDURE FillCOval(r: Rect;pp: PixPatHandle);
013948      INLINE $AA0F;
013949  PROCEDURE FillCRoundRect(r: Rect;ovalWidth: INTEGER;ovalHeight: INTEGER;
013950      pp: PixPatHandle);
013951      INLINE $AA10;
013952  PROCEDURE FillCArc(r: Rect;startAngle: INTEGER;arcAngle: INTEGER;pp: PixPatHandle);
013953      INLINE $AA11;
013954  PROCEDURE FillCRgn(rgn: RgnHandle;pp: PixPatHandle);
013955      INLINE $AA12;
013956  PROCEDURE FillCPoly(poly: PolyHandle;pp: PixPatHandle);
013957      INLINE $AA13;
013958  PROCEDURE RGBForeColor(color: RGBColor);
013959      INLINE $AA14;
013960  PROCEDURE RGBBackColor(color: RGBColor);
013961      INLINE $AA15;
013962  PROCEDURE SetCPixel(h: INTEGER;v: INTEGER;cPix: RGBColor);
013963      INLINE $AA16;
013964  PROCEDURE SetPortPix(pm: PixMapHandle);
013965      INLINE $AA06;
013966  PROCEDURE GetCPixel(h: INTEGER;v: INTEGER;VAR cPix: RGBColor);
013967      INLINE $AA17;
013968  PROCEDURE GetForeColor(VAR color: RGBColor);
013969      INLINE $AA19;
013970  PROCEDURE GetBackColor(VAR color: RGBColor);
013971      INLINE $AA1A;
013972  PROCEDURE SeedCFill(srcBits: BitMap;dstBits: BitMap;srcRect: Rect;dstRect: Rect;
013973      seedH: INTEGER;seedV: INTEGER;matchProc: ProcPtr;matchData: LONGINT);
013974      INLINE $AA50;
013975  PROCEDURE CalcCMask(srcBits: BitMap;dstBits: BitMap;srcRect: Rect;dstRect: Rect;
013976      seedRGB: RGBColor;matchProc: ProcPtr;matchData: LONGINT);
013977      INLINE $AA4F;
013978  FUNCTION OpenCPicture(newHeader: OpenCPicParams): PicHandle;
```

```
013979      INLINE $AA20;
013980  PROCEDURE OpColor(color: RGBColor);
013981      INLINE $AA21;
013982  PROCEDURE HiliteColor(color: RGBColor);
013983      INLINE $AA22;
013984  PROCEDURE DisposCTable(cTable: CTabHandle);
013985      INLINE $AA24;
013986  PROCEDURE DisposeCTable(cTable: CTabHandle);
013987      INLINE $AA24;
013988  FUNCTION GetCTable(ctID: INTEGER): CTabHandle;
013989      INLINE $AA18;
013990  FUNCTION GetCCursor(crsrID: INTEGER): CCrsrHandle;
013991      INLINE $AA1B;
013992  PROCEDURE SetCCursor(cCrsr: CCrsrHandle);
013993      INLINE $AA1C;
013994  PROCEDURE AllocCursor;
013995      INLINE $AA1D;
013996  PROCEDURE DisposCCursor(cCrsr: CCrsrHandle);
013997      INLINE $AA26;
013998  PROCEDURE DisposeCCursor(cCrsr: CCrsrHandle);
013999      INLINE $AA26;
014000  FUNCTION GetCIcon(iconID: INTEGER): CIconHandle;
014001      INLINE $AA1E;
014002  PROCEDURE PlotCIcon(theRect: Rect;theIcon: CIconHandle);
014003      INLINE $AA1F;
014004  PROCEDURE DisposCIcon(theIcon: CIconHandle);
014005      INLINE $AA25;
014006  PROCEDURE DisposeCIcon(theIcon: CIconHandle);
014007      INLINE $AA25;
014008  PROCEDURE SetStdCProcs(VAR procs: CQDProcs);
014009      INLINE $AA4E;
014010  PROCEDURE CharExtra(extra: Fixed);
014011      INLINE $AA23;
014012  FUNCTION GetMaxDevice(globalRect: Rect): GDHandle;
014013      INLINE $AA27;
014014  FUNCTION GetCTSeed: LONGINT;
014015      INLINE $AA28;
014016  FUNCTION GetDeviceList: GDHandle;
014017      INLINE $AA29;
014018  FUNCTION GetMainDevice: GDHandle;
014019      INLINE $AA2A;
014020  FUNCTION GetNextDevice(curDevice: GDHandle): GDHandle;
014021      INLINE $AA2B;
014022  FUNCTION TestDeviceAttribute(gdh: GDHandle;attribute: INTEGER): BOOLEAN;
014023      INLINE $AA2C;
014024  PROCEDURE SetDeviceAttribute(gdh: GDHandle;attribute: INTEGER;value: BOOLEAN);
014025      INLINE $AA2D;
014026  PROCEDURE InitGDevice(qdRefNum: INTEGER;mode: LONGINT;gdh: GDHandle);
014027      INLINE $AA2E;
014028  FUNCTION NewGDevice(refNum: INTEGER;mode: LONGINT): GDHandle;
014029      INLINE $AA2F;
014030  PROCEDURE DisposGDevice(gdh: GDHandle);
014031      INLINE $AA30;
014032  PROCEDURE DisposeGDevice(gdh: GDHandle);
014033      INLINE $AA30;
014034  PROCEDURE SetGDevice(gd: GDHandle);
```

```
014035      INLINE $AA31;
014036  FUNCTION GetGDevice: GDHandle;
014037      INLINE $AA32;
014038  FUNCTION Color2Index(myColor: RGBColor): LONGINT;
014039      INLINE $AA33;
014040  PROCEDURE Index2Color(index: LONGINT;VAR aColor: RGBColor);
014041      INLINE $AA34;
014042  PROCEDURE InvertColor(VAR myColor: RGBColor);
014043      INLINE $AA35;
014044  FUNCTION RealColor(color: RGBColor): BOOLEAN;
014045      INLINE $AA36;
014046  PROCEDURE GetSubTable(myColors: CTabHandle;iTabRes: INTEGER;targetTbl: CTabHandle);
014047      INLINE $AA37;
014048  PROCEDURE MakeITable(cTabH: CTabHandle;iTabH: ITabHandle;res: INTEGER);
014049      INLINE $AA39;
014050  PROCEDURE AddSearch(searchProc: ProcPtr);
014051      INLINE $AA3A;
014052  PROCEDURE AddComp(compProc: ProcPtr);
014053      INLINE $AA3B;
014054  PROCEDURE DelSearch(searchProc: ProcPtr);
014055      INLINE $AA4C;
014056  PROCEDURE DelComp(compProc: ProcPtr);
014057      INLINE $AA4D;
014058  PROCEDURE SubPt(src: Point;VAR dst: Point);
014059      INLINE $A87F;
014060  PROCEDURE SetClientID(id: INTEGER);
014061      INLINE $AA3C;
014062  PROCEDURE ProtectEntry(index: INTEGER;protect: BOOLEAN);
014063      INLINE $AA3D;
014064  PROCEDURE ReserveEntry(index: INTEGER;reserve: BOOLEAN);
014065      INLINE $AA3E;
014066  PROCEDURE SetEntries(start: INTEGER;count: INTEGER;aTable: CSpecArray);
014067      INLINE $AA3F;
014068  PROCEDURE SaveEntries(srcTable: CTabHandle;resultTable: CTabHandle;VAR selection: ReqListRec);
014069      INLINE $AA49;
014070  PROCEDURE RestoreEntries(srcTable: CTabHandle;dstTable: CTabHandle;VAR selection: ReqListRec);
014071      INLINE $AA4A;
014072  FUNCTION QDError: INTEGER;
014073      INLINE $AA40;
014074  PROCEDURE CopyDeepMask(srcBits: BitMap;maskBits: BitMap;dstBits: BitMap;
014075      srcRect: Rect;maskRect: Rect;dstRect: Rect;mode: INTEGER;maskRgn: RgnHandle);
014076      INLINE $AA51;
014077  PROCEDURE DeviceLoop(drawingRgn: RgnHandle;drawingProc: DeviceLoopDrawingProcPtr;
014078      userData: LONGINT;flags: DeviceLoopFlags);
014079      INLINE $ABCA;
014080  FUNCTION GetMaskTable: Ptr;
014081      INLINE $A836,$2E88;
014082
014083
014084  {$ENDC}    { UsingQuickdraw }
014085
014086  {$IFC NOT UsingIncludes}
014087      END.
014088  {$ENDC}
014089
014090
```

```
014091  ### END OF FILE Quickdraw.p
014092
```

```
014093
014094  ##########################################################################
014095  ### FILE: Resources.p
014096  ##########################################################################
014097
014098
014099  {
014100  Created: Thursday, September 12, 1991 at 2:50 PM
014101   Resources.p
014102   Pascal Interface to the Macintosh Libraries
014103
014104    Copyright Apple Computer, Inc. 1985-1991
014105    All rights reserved
014106  }
014107
014108
014109  {$IFC UNDEFINED UsingIncludes}
014110  {$SETC UsingIncludes := 0}
014111  {$ENDC}
014112
014113  {$IFC NOT UsingIncludes}
014114   UNIT Resources;
014115   INTERFACE
014116  {$ENDC}
014117
014118  {$IFC UNDEFINED UsingResources}
014119  {$SETC UsingResources := 1}
014120
014121  {$I+}
014122  {$SETC ResourcesIncludes := UsingIncludes}
014123  {$SETC UsingIncludes := 1}
014124  {$IFC UNDEFINED UsingTypes}
014125  {$I $$Shell(PInterfaces)Types.p}
014126  {$ENDC}
014127  {$IFC UNDEFINED UsingFiles}
014128  {$I $$Shell(PInterfaces)Files.p}
014129  {$ENDC}
014130  {$SETC UsingIncludes := ResourcesIncludes}
014131
014132  CONST
014133  resSysHeap = 64;    {System or application heap?}
014134  resPurgeable = 32;  {Purgeable resource?}
014135  resLocked = 16;     {Load it in locked?}
014136  resProtected = 8;   {Protected?}
014137  resPreload = 4;     {Load in on OpenResFile?}
014138  resChanged = 2;     {Resource changed?}
014139  mapReadOnly = 128;  {Resource file read-only}
014140  mapCompact = 64;    {Compact resource file}
014141  mapChanged = 32;    {Write map out at updat}
014142
014143  { Values for setting RomMapInsert and TmpResLoad }
014144  mapTrue = $FFFF;    {insert ROM map w/ TmpResLoad = TRUE.}
014145  mapFalse = $FF00;   {insert ROM map w/ TmpResLoad = FALSE.}
014146
014147  FUNCTION InitResources: INTEGER;
014148   INLINE $A995;
```

```
014149  PROCEDURE RsrcZoneInit;
014150   INLINE $A996;
014151  PROCEDURE CloseResFile(refNum: INTEGER);
014152   INLINE $A99A;
014153  FUNCTION ResError: INTEGER;
014154   INLINE $A9AF;
014155  FUNCTION CurResFile: INTEGER;
014156   INLINE $A994;
014157  FUNCTION HomeResFile(theResource: Handle): INTEGER;
014158   INLINE $A9A4;
014159  PROCEDURE CreateResFile(fileName: Str255);
014160   INLINE $A9B1;
014161  FUNCTION OpenResFile(fileName: Str255): INTEGER;
014162   INLINE $A997;
014163  PROCEDURE UseResFile(refNum: INTEGER);
014164   INLINE $A998;
014165  FUNCTION CountTypes: INTEGER;
014166   INLINE $A99E;
014167  FUNCTION Count1Types: INTEGER;
014168   INLINE $A81C;
014169  PROCEDURE GetIndType(VAR theType: ResType;index: INTEGER);
014170   INLINE $A99F;
014171  PROCEDURE Get1IndType(VAR theType: ResType;index: INTEGER);
014172   INLINE $A80F;
014173  PROCEDURE SetResLoad(load: BOOLEAN);
014174   INLINE $A99B;
014175  FUNCTION CountResources(theType: ResType): INTEGER;
014176   INLINE $A99C;
014177  FUNCTION Count1Resources(theType: ResType): INTEGER;
014178   INLINE $A80D;
014179  FUNCTION GetIndResource(theType: ResType;index: INTEGER): Handle;
014180   INLINE $A99D;
014181  FUNCTION Get1IndResource(theType: ResType;index: INTEGER): Handle;
014182   INLINE $A80E;
014183  FUNCTION GetResource(theType: ResType;theID: INTEGER): Handle;
014184   INLINE $A9A0;
014185  FUNCTION Get1Resource(theType: ResType;theID: INTEGER): Handle;
014186   INLINE $A81F;
014187  FUNCTION GetNamedResource(theType: ResType;name: Str255): Handle;
014188   INLINE $A9A1;
014189  FUNCTION Get1NamedResource(theType: ResType;name: Str255): Handle;
014190   INLINE $A820;
014191  PROCEDURE LoadResource(theResource: Handle);
014192   INLINE $A9A2;
014193  PROCEDURE ReleaseResource(theResource: Handle);
014194   INLINE $A9A3;
014195  PROCEDURE DetachResource(theResource: Handle);
014196   INLINE $A992;
014197  FUNCTION UniqueID(theType: ResType): INTEGER;
014198   INLINE $A9C1;
014199  FUNCTION Unique1ID(theType: ResType): INTEGER;
014200   INLINE $A810;
014201  FUNCTION GetResAttrs(theResource: Handle): INTEGER;
014202   INLINE $A9A6;
014203  PROCEDURE GetResInfo(theResource: Handle;VAR theID: INTEGER;VAR theType: ResType;
014204   VAR name: Str255);
```

```
014205   INLINE $A9A8;
014206  PROCEDURE SetResInfo(theResource: Handle;theID: INTEGER;name: Str255);
014207   INLINE $A9A9;
014208  PROCEDURE AddResource(theResource: Handle;theType: ResType;theID: INTEGER;
014209   name: Str255);
014210   INLINE $A9AB;
014211  FUNCTION SizeResource(theResource: Handle): LONGINT;
014212   INLINE $A9A5;
014213  FUNCTION MaxSizeRsrc(theResource: Handle): LONGINT;
014214   INLINE $A821;
014215  FUNCTION RsrcMapEntry(theResource: Handle): LONGINT;
014216   INLINE $A9C5;
014217  PROCEDURE SetResAttrs(theResource: Handle;attrs: INTEGER);
014218   INLINE $A9A7;
014219  PROCEDURE ChangedResource(theResource: Handle);
014220   INLINE $A9AA;
014221  PROCEDURE RmveResource(theResource: Handle);
014222   INLINE $A9AD;
014223  PROCEDURE UpdateResFile(refNum: INTEGER);
014224   INLINE $A999;
014225  PROCEDURE WriteResource(theResource: Handle);
014226   INLINE $A9B0;
014227  PROCEDURE SetResPurge(install: BOOLEAN);
014228   INLINE $A993;
014229  FUNCTION GetResFileAttrs(refNum: INTEGER): INTEGER;
014230   INLINE $A9F6;
014231  PROCEDURE SetResFileAttrs(refNum: INTEGER;attrs: INTEGER);
014232   INLINE $A9F7;
014233  FUNCTION OpenRFPerm(fileName: Str255;vRefNum: INTEGER;permission: SignedByte): INTEGER;
014234   INLINE $A9C4;
014235  FUNCTION RGetResource(theType: ResType;theID: INTEGER): Handle;
014236   INLINE $A80C;
014237  FUNCTION HOpenResFile(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255;
014238   permission: SignedByte): INTEGER;
014239  {$IFC SystemSevenOrLater }
014240   INLINE $A81A;
014241  {$ENDC}
014242  PROCEDURE HCreateResFile(vRefNum: INTEGER;dirID: LONGINT;fileName: Str255);
014243  {$IFC SystemSevenOrLater }
014244   INLINE $A81B;
014245  {$ENDC}
014246
014247  FUNCTION FSpOpenResFile(spec: FSSpec;permission: SignedByte): INTEGER;
014248   INLINE $303C,$000D,$AA52;
014249  PROCEDURE FSpCreateResFile(spec: FSSpec;creator: OSType;fileType: OSType;
014250   scriptTag: ScriptCode);
014251   INLINE $303C,$000E,$AA52;
014252
014253  {  partial resource calls  }
014254  PROCEDURE ReadPartialResource(theResource: Handle;offset: LONGINT;buffer: UNIV Ptr;
014255   count: LONGINT);
014256   INLINE $7001,$A822;
014257  PROCEDURE WritePartialResource(theResource: Handle;offset: LONGINT;buffer: UNIV Ptr;
014258   count: LONGINT);
014259   INLINE $7002,$A822;
014260  PROCEDURE SetResourceSize(theResource: Handle;newSize: LONGINT);
```

```
014261   INLINE $7003,$A822;
014262
014263
014264
014265  {$ENDC} { UsingResources }
014266
014267  {$IFC NOT UsingIncludes}
014268   END.
014269  {$ENDC}
014270
014271
014272  ### END OF FILE Resources.p
014273
```

```
014274
014275  ##########################################################################
014276  ### FILE: Retrace.p
014277  ##########################################################################
014278
014279  {
014280  Created: Sunday, January 6, 1991 at 11:06 PM
014281      Retrace.p
014282      Pascal Interface to the Macintosh Libraries
014283
014284          Copyright Apple Computer, Inc.    1985-1990
014285          All rights reserved
014286  }
014287
014288
014289  {$IFC UNDEFINED UsingIncludes}
014290  {$SETC UsingIncludes := 0}
014291  {$ENDC}
014292
014293  {$IFC NOT UsingIncludes}
014294      UNIT Retrace;
014295      INTERFACE
014296  {$ENDC}
014297
014298  {$IFC UNDEFINED UsingRetrace}
014299  {$SETC UsingRetrace := 1}
014300
014301  {$I+}
014302  {$SETC RetraceIncludes := UsingIncludes}
014303  {$SETC UsingIncludes := 1}
014304  {$IFC UNDEFINED UsingTypes}
014305  {$I $$Shell(PInterfaces)Types.p}
014306  {$ENDC}
014307  {$IFC UNDEFINED UsingOSUtils}
014308  {$I $$Shell(PInterfaces)OSUtils.p}
014309  {$ENDC}
014310  {$SETC UsingIncludes := RetraceIncludes}
014311
014312  FUNCTION GetVBLQHdr: QHdrPtr;
014313      INLINE $2EBC,$0000,$0160;
014314  FUNCTION SlotVInstall(vblBlockPtr: QElemPtr;theSlot: INTEGER): OSErr;
014315      INLINE $301F,$205F,$A06F,$3E80;
014316  FUNCTION SlotVRemove(vblBlockPtr: QElemPtr;theSlot: INTEGER): OSErr;
014317      INLINE $301F,$205F,$A070,$3E80;
014318  FUNCTION AttachVBL(theSlot: INTEGER): OSErr;
014319      INLINE $301F,$A071,$3E80;
014320  FUNCTION DoVBLTask(theSlot: INTEGER): OSErr;
014321      INLINE $301F,$A072,$3E80;
014322  FUNCTION VInstall(vblTaskPtr: QElemPtr): OSErr;
014323      INLINE $205F,$A033,$3E80;
014324  FUNCTION VRemove(vblTaskPtr: QElemPtr): OSErr;
014325      INLINE $205F,$A034,$3E80;
014326
014327
014328  {$ENDC}    { UsingRetrace }
014329
```

```
014330  {$IFC NOT UsingIncludes}
014331      END.
014332  {$ENDC}
014333
014334
014335  ### END OF FILE Retrace.p
014336
```

```
014337
014338  ##########################################################################
014339  ### FILE: ROMDefs.p
014340  ##########################################################################
014341
014342  {
014343  Created: Sunday, January 6, 1991 at 11:06 PM
014344      ROMDefs.p
014345      Pascal Interface to the Macintosh Libraries
014346
014347          Copyright Apple Computer, Inc.    1986-1990
014348          All rights reserved
014349  }
014350
014351
014352  {$IFC UNDEFINED UsingIncludes}
014353  {$SETC UsingIncludes := 0}
014354  {$ENDC}
014355
014356  {$IFC NOT UsingIncludes}
014357      UNIT ROMDefs;
014358      INTERFACE
014359  {$ENDC}
014360
014361  {$IFC UNDEFINED UsingROMDefs}
014362  {$SETC UsingROMDefs := 1}
014363
014364
014365  CONST
014366  appleFormat = 1;                        {Format of Declaration Data (IEEE will assign real value)}
014367  romRevision = 1;                        {Revision of Declaration Data Format}
014368  testPattern = 1519594439;               {FHeader long word test pattern}
014369
014370  sCodeRev = 2;                           {Revision of code (For sExec)}
014371  sCPU68000 = 1;                          {CPU type = 68000}
014372  sCPU68020 = 2;                          {CPU type = 68020}
014373  sCPU68030 = 3;                          {CPU type = 68030}
014374  sCPU68040 = 4;                          {CPU type = 68040}
014375  sMacOS68000 = 1;                        {Mac OS, CPU type = 68000}
014376  sMacOS68020 = 2;                        {Mac OS, CPU type = 68020}
014377  sMacOS68030 = 3;                        {Mac OS, CPU type = 68030}
014378  sMacOS68040 = 4;                        {Mac OS, CPU type = 68040}
014379
014380  board = 0;                              {Board sResource - Required on all boards}
014381  displayVideoAppleTFB = 16843009;        {Video with Apple parameters for TFB card.}
014382  displayVideoAppleGM = 16843010;         {Video with Apple parameters for GM card.}
014383  networkEtherNetApple3Com = 33620225;    {Ethernet with apple parameters for 3-Comm card.}
014384  testSimpleAppleAny = -2147417856;       {A simple test sResource.}
014385  endOfList = 255;                        {End of list}
014386  defaultTO = 100;                        {100 retries.}
014387
014388  sRsrcType = 1;                          {Type of sResource}
014389  sRsrcName = 2;                          {Name of sResource}
014390  sRsrcIcon = 3;                          {Icon}
014391  sRsrcDrvrDir = 4;                       {Driver directory}
014392  sRsrcLoadDir = 5;                       {Load directory}
```

```
014393  sRsrcBootRec = 6;                       {sBoot record}
014394  sRsrcFlags = 7;                         {sResource Flags}
014395  sRsrcHWDevId = 8;                       {Hardware Device Id}
014396  minorBaseOS = 10;                       {Offset to base of sResource in minor space.}
014397  minorLength = 11;                       {Length of sResource's address space in standard slot space.}
014398  majorBaseOS = 12;                       {Offset to base of sResource in Major space.}
014399  majorLength = 13;                       {Length of sResource in super slot space.}
014400  sRsrccicn = 15;                         {Color icon}
014401  sRsrcicl8 = 16;                         {8-bit (indexed) icon}
014402  sRsrcicl4 = 17;                         {4-bit (indexed) icon}
014403  sGammaDir = 64;                         {sGamma directory}
014404  sDRVRDir = 16;                          {sDriver directory}
014405
014406  drSwApple = 1;                          {To ask for or define an Apple-compatible SW device.}
014407  drHwTFB = 1;                            {HW ID for the TFB (original Mac II) video card.}
014408  drHw3Com = 1;                           {HW ID for the Apple EtherTalk card.}
014409  drHwBSC = 3;
014410  catBoard = 1;                           {Category for board types.}
014411  catTest = 2;                            {Category for test types -- not used much.}
014412  catDisplay = 3;                         {Category for display (video) cards.}
014413  catNetwork = 4;                         {Category for Networking cards.}
014414
014415  boardId = 32;                           {Board Id}
014416  pRAMInitData = 33;                      {sPRAM init data}
014417  primaryInit = 34;                       {Primary init record}
014418  timeOutConst = 35;                      {Time out constant}
014419  vendorInfo = 36;                        {Vendor information List. See Vendor List, below}
014420  boardFlags = 37;                        {Board Flags}
014421  secondaryInit = 38;                     {Secondary init record/code}
014422  sRsrcVidNames = 65;                     {Video mode name directory}
014423
014424  vendorId = 1;                           {Vendor Id}
014425  serialNum = 2;                          {Serial number}
014426  revLevel = 3;                           {Revision level}
014427  partNum = 4;                            {Part number}
014428  date = 5;                               {Last revision date of the card}
014429
014430  typeBoard = 0;                          {Type for board types.}
014431  typeApple = 1;
014432  typeVideo = 1;                          {Type for video types.}
014433  typeEtherNet = 1;                       {Type for ethernet types.}
014434  testByte = 32;                          {Test byte.}
014435  testWord = 33;                          {0021}
014436  testLong = 34;                          {Test Long.}
014437  testString = 35;                        {Test String.}
014438
014439
014440  {$ENDC}    { UsingROMDefs }
014441
014442  {$IFC NOT UsingIncludes}
014443      END.
014444  {$ENDC}
014445
014446
014447  ### END OF FILE ROMDefs.p
014448
```

```
014449
014450  ############################################################################
014451  ### FILE: RTLib.p
014452  ############################################################################
014453
014454
014455  {
014456  Created: Friday, August 2, 1991 at 11:20 PM
014457   RTLib.p
014458   Pascal Interface to the Macintosh Libraries
014459
014460    Copyright Apple Computer, Inc. 1990-1991
014461    All rights reserved
014462  }
014463
014464
014465  {$IFC UNDEFINED UsingIncludes}
014466  {$SETC UsingIncludes := 0}
014467  {$ENDC}
014468
014469  {$IFC NOT UsingIncludes}
014470   UNIT RTLib;
014471   INTERFACE
014472  {$ENDC}
014473
014474  {$IFC UNDEFINED UsingRTLib}
014475  {$SETC UsingRTLib := 1}
014476
014477  {$I+}
014478  {$SETC RTLibIncludes := UsingIncludes}
014479  {$SETC UsingIncludes := 1}
014480  {$IFC UNDEFINED UsingTypes}
014481  {$I $$Shell(PInterfaces)Types.p}
014482  {$ENDC}
014483  {$SETC UsingIncludes := RTLibIncludes}
014484
014485
014486  CONST
014487      {
014488              Error Codes
014489      }
014490      eRTNoErr                = 0;
014491      eRTBadVersion       = 2;
014492      eRTInvalidOp        = 4;
014493      eRTInvalidJTPtr = 6;
014494
014495      {
014496              Action Codes
014497      }
014498      kRTSysErr   = 0;
014499      kRTRetry        = 1;
014500      kRTContinue = 2;
014501
014502      {
014503              Runtime Operations
014504      }
```

```
014505      kRTGetVersion           = 10;
014506      kRTGetVersionA5     = 11;
014507      kRTGetJTAddress     = 12;
014508      kRTGetJTAddressA5       = 13;
014509      kRTSetPreLoad           = 14;
014510      kRTSetPreLoadA5     = 15;
014511      kRTSetSegLoadErr        = 16;
014512      kRTSetSegLoadErrA5  = 17;
014513      kRTSetPostLoad          = 18;
014514      kRTSetPostLoadA5        = 19;
014515      kRTSetPreUnload     = 20;
014516      kRTSetPreUnloadA5       = 21;
014517      kRTPreLaunch            = 22;
014518      kRTPostLaunch           = 23;
014519
014520      {
014521              Version Definitions
014522      }
014523      kVERSION32BIT           = $FFFF;
014524      kVERSION16BIT           = $0000;
014525
014526
014527  TYPE
014528      {
014529              RTState Definition
014530      }
014531      RTState = RECORD
014532          fVersion:   INTEGER;                        { run-time version }
014533          fSP:            Ptr;                            { SP: &-of user return address }
014534          fJTAddr:        Ptr;                            { PC: &-of called jump table entry }
014535          fRegisters: Array[0..14] of LONGINT;    { registers D0-D7 and A0-A6 when }
014536                                                          {       _LoadSeg was called }
014537          fSegNo:     INTEGER;                        {   segment number }
014538          fSegType:   ResType;                        {   segment type (normally 'CODE') }
014539          fSegSize:   LONGINT;                        {   segment size }
014540          fSegInCore: BOOLEAN;                        {   true if segment is in memory }
014541          fReserved1: BOOLEAN;                        { (reserved for future use) }
014542          fOSErr:     OSErr;                      {   error number }
014543          fReserved2: LONGINT;                        { (reserved for future use) }
014544      END;
014545
014546      RTStatePtr = ^RTState;
014547
014548      {
014549              Runtime Parameter Block
014550      }
014551
014552      RTParam = (RTGetVersionParam, RTGetJTAddrParam, RTSetSegLoadParam);
014553
014554      RTPB = RECORD
014555          fOperation:     INTEGER;                    { operation }
014556          fA5:                Ptr;                        { A5-world }
014557          CASE RTParam OF
014558              RTGetVersionParam:
014559                  (fVersion:  INTEGER);               { run-time version (returned) }
014560
```

```
014561              RTGetJTAddrParam:
014562                  (fJTAddr:   Ptr;                        { ptr to jt entry }
014563                  fCodeAddr:  Ptr);                       { code address w/i jt entry (returned) }
014564
014565              RTSetSegLoadParam:
014566                  (fUserHdlr:     Ptr;                    { ptr to user handler }
014567                  fOldUserHdlr:   Ptr);                   { ptr to old user handler (returned) }
014568          END;
014569
014570      RTPBPtr = ^RTPB;
014571
014572
014573  FUNCTION Runtime (runtime_parms: RTPBPtr): OSErr;
014574
014575
014576  {$ENDC}    { UsingRTLib }
014577
014578  {$IFC NOT UsingIncludes}
014579      END.
014580  {$ENDC}
014581
014582
014583  ### END OF FILE RTLib.p
014584
```

```
014585
014586  ############################################################################
014587  ### FILE: SANE.p
014588  ############################################################################
014589
014590  {
014591  Created: Friday, September 15, 1989 at 5:01 PM
014592      SANE.p
014593      Pascal Interface to the Macintosh Libraries
014594
014595      Copyright Apple Computer, Inc.  1985-1991
014596      All rights reserved
014597  }
014598
014599
014600  {$IFC UNDEFINED UsingIncludes}
014601  {$SETC UsingIncludes := 0}
014602  {$ENDC}
014603
014604  {$IFC NOT UsingIncludes}
014605      UNIT SANE;
014606      INTERFACE
014607  {$ENDC}
014608
014609  {$IFC UNDEFINED UsingSANE}
014610  {$SETC UsingSANE := 1}
014611
014612  {$I+}
014613  {$SETC SANEIncludes := UsingIncludes}
014614  {$SETC UsingIncludes := 1}
014615  {$IFC UNDEFINED UsingTypes}
014616  {$I $$Shell(PInterfaces)Types.p}
014617  {$ENDC}
014618  {$SETC UsingIncludes := SANEIncludes}
014619
014620  { Elems881 mode set by -d Elems881=true on Pascal command line }
014621
014622  {$IFC UNDEFINED Elems881}
014623  {$SETC Elems881 = FALSE}
014624  {$ENDC}
014625
014626
014627  CONST
014628  {$IFC OPTION(MC68881)}
014629
014630  {*====================================================================*
014631   *  The interface specific to the MC68881 SANE library  *
014632   *====================================================================*}
014633
014634  Inexact = 8;
014635  DivByZero = 16;
014636  Underflow = 32;
014637  Overflow = 64;
014638  Invalid = 128;
014639  CurInex1 = 256;
014640  CurInex2 = 512;
```

```
014641  CurDivByZero = 1024;
014642  CurUnderflow = 2048;
014643  CurOverflow = 4096;
014644  CurOpError = 8192;
014645  CurSigNaN = 16384;
014646  CurBSonUnor = 32768;
014647
014648
014649  {$ELSEC}
014650
014651  {*======================================================================*
014652   *  The interface specific to the software SANE library    *
014653   *======================================================================*}
014654
014655  Invalid = 1;
014656  Underflow = 2;
014657  Overflow = 4;
014658  DivByZero = 8;
014659  Inexact = 16;
014660  IEEEDefaultEnv = 0;     {IEEE-default floating-point environment constant}
014661
014662
014663  {$ENDC}
014664
014665  {*======================================================================*
014666   *  The common interface for the SANE library    *
014667   *======================================================================*}
014668
014669  DecStrLen = 255;
014670  SigDigLen = 20;         {for 68K; use 28 in 6502 SANE}
014671
014672
014673  TYPE
014674
014675  RelOp = (GreaterThan,LessThan,EqualTo,Unordered);
014676
014677  NumClass = (SNaN,QNaN,Infinite,ZeroNum,NormalNum,DenormalNum);
014678
014679  RoundDir = (ToNearest,Upward,Downward,TowardZero);
014680
014681  RoundPre = (ExtPrecision,DblPrecision,RealPrecision);
014682
014683  DecimalKind = (FloatDecimal,FixedDecimal);
014684
014685  {$IFC OPTION(MC68881)}
014686
014687  {*======================================================================*
014688   *  The interface specific to the MC68881 SANE library  *
014689   *======================================================================*}
014690  Exception = LONGINT;
014691
014692  Environment = RECORD
014693      FPCR: LONGINT;
014694      FPSR: LONGINT;
014695      END;
014696
```

```
014697  TrapVector = RECORD
014698      Unordered: LONGINT;
014699      Inexact: LONGINT;
014700      DivByZero: LONGINT;
014701      Underflow: LONGINT;
014702      OpError: LONGINT;
014703      Overflow: LONGINT;
014704      SigNaN: LONGINT;
014705      END;
014706
014707  {$ELSEC}
014708
014709  {*=========================================================================*
014710  *  The interface specific to the software SANE library    *
014711  *=========================================================================*}
014712
014713  Exception = INTEGER;
014714
014715  Environment = INTEGER;
014716
014717  Extended96 = ARRAY [0..5] OF INTEGER;
014718
014719  MiscHaltInfo = RECORD
014720      HaltExceptions: INTEGER;
014721      PendingCCR: INTEGER;
014722      PendingD0: LONGINT;
014723      END;
014724
014725  {$ENDC}
014726
014727  {*=========================================================================*
014728  *  The common interface for the SANE library    *
014729  *=========================================================================*}
014730
014731  DecStr = STRING[DecStrLen];
014732
014733  DecForm = RECORD
014734      style: DecimalKind;
014735      digits: INTEGER;
014736      END;
014737
014738  Decimal = RECORD
014739      sgn: 0..1;
014740      exp: INTEGER;
014741      sig: STRING[SigDigLen];
014742      END;
014743
014744  CStrPtr = ^CHAR;
014745
014746
014747
014748  {$IFC OPTION(MC68881)}
014749
014750  { return IEEE default environment }
014751  FUNCTION IEEEDefaultEnv: environment;
014752  PROCEDURE SetTrapVector(Traps: trapvector);
```

```
014753  PROCEDURE GetTrapVector(VAR Traps: trapvector);
014754  FUNCTION X96toX80(x: Extended): extended80;
014755  FUNCTION X80toX96(x: extended80): Extended;
014756  {$IFC Elems881 = false}
014757
014758  { sine }FUNCTION Sin(x: Extended): Extended;
014759  FUNCTION Cos(x: Extended): Extended;
014760  FUNCTION ArcTan(x: Extended): Extended;
014761  FUNCTION Exp(x: Extended): Extended;
014762  FUNCTION Ln(x: Extended): Extended;
014763  FUNCTION Log2(x: Extended): Extended;
014764  FUNCTION Ln1(x: Extended): Extended;
014765  FUNCTION Exp2(x: Extended): Extended;
014766  FUNCTION Exp1(x: Extended): Extended;
014767  FUNCTION Tan(x: Extended): Extended;
014768  {$ENDC}
014769
014770  {$ELSEC}
014771
014772  { return halt vector }FUNCTION GetHaltVector: LONGINT;
014773  PROCEDURE SetHaltVector(v: LONGINT);
014774  FUNCTION X96toX80(x: Extended96): Extended;
014775  FUNCTION X80toX96(x: Extended): Extended96;
014776  FUNCTION Log2(x: Extended): Extended;
014777  FUNCTION Ln1(x: Extended): Extended;
014778  FUNCTION Exp2(x: Extended): Extended;
014779  FUNCTION Exp1(x: Extended): Extended;
014780  FUNCTION Tan(x: Extended): Extended;
014781  {$ENDC}
014782
014783
014784  {*====================================================================*
014785  *  The common interface for the SANE library    *
014786  *====================================================================*}
014787
014788  {-------------------------------------------------
014789  * Conversions between numeric binary types.
014790  -------------------------------------------------}
014791
014792  FUNCTION Num2Integer(x: Extended): INTEGER;
014793  FUNCTION Num2Longint(x: Extended): LONGINT;
014794  FUNCTION Num2Real(x: Extended): real;
014795  FUNCTION Num2Double(x: Extended): DOUBLE;
014796  FUNCTION Num2Extended(x: Extended): Extended;
014797  FUNCTION Num2Comp(x: Extended): Comp;
014798  PROCEDURE Num2Dec(f: decform;x: Extended;VAR d: decimal);
014799  FUNCTION Dec2Num(d: decimal): Extended;
014800  PROCEDURE Num2Str(f: decform;x: Extended;VAR s: DecStr);
014801  FUNCTION Str2Num(s: DecStr): Extended;
014802  PROCEDURE Str2Dec(s: DecStr;VAR Index: INTEGER;VAR d: decimal;VAR ValidPrefix: BOOLEAN);
014803  PROCEDURE CStr2Dec(s: CStrPtr;VAR Index: INTEGER;VAR d: decimal;VAR ValidPrefix: BOOLEAN);
014804  PROCEDURE Dec2Str(f: decform;d: decimal;VAR s: DecStr);
014805  FUNCTION Remainder(x: Extended;y: Extended;VAR quo: INTEGER): Extended;
014806  FUNCTION Rint(x: Extended): Extended;
014807  FUNCTION Scalb(n: INTEGER;x: Extended): Extended;
014808  FUNCTION Logb(x: Extended): Extended;
```

```
014809  FUNCTION CopySign(x: Extended;y: Extended): Extended;
014810  FUNCTION NextReal(x: real;y: real): real;
014811  FUNCTION NextDouble(x: DOUBLE;y: DOUBLE): DOUBLE;
014812  FUNCTION NextExtended(x: Extended;y: Extended): Extended;
014813  FUNCTION XpwrI(x: Extended;i: INTEGER): Extended;
014814  FUNCTION XpwrY(x: Extended;y: Extended): Extended;
014815  FUNCTION Compound(r: Extended;n: Extended): Extended;
014816  FUNCTION Annuity(r: Extended;n: Extended): Extended;
014817  FUNCTION RandomX(VAR x: Extended): Extended;
014818  FUNCTION ClassReal(x: real): NumClass;
014819  FUNCTION ClassDouble(x: DOUBLE): NumClass;
014820  FUNCTION ClassComp(x: Comp): NumClass;
014821  FUNCTION ClassExtended(x: Extended): NumClass;
014822  FUNCTION SignNum(x: Extended): INTEGER;
014823  FUNCTION NAN(i: INTEGER): Extended;
014824  PROCEDURE SetException(e: Exception;b: BOOLEAN);
014825  FUNCTION TestException(e: Exception): BOOLEAN;
014826  PROCEDURE SetHalt(e: Exception;b: BOOLEAN);
014827  FUNCTION TestHalt(e: Exception): BOOLEAN;
014828  PROCEDURE SetRound(r: RoundDir);
014829  FUNCTION GetRound: RoundDir;
014830  PROCEDURE SetPrecision(p: RoundPre);
014831  FUNCTION GetPrecision: RoundPre;
014832  PROCEDURE SetEnvironment(e: environment);
014833  PROCEDURE GetEnvironment(VAR e: environment);
014834  PROCEDURE ProcEntry(VAR e: environment);
014835  PROCEDURE ProcExit(e: environment);
014836  FUNCTION Relation(x: Extended;y: Extended): RelOp;
014837
014838  {$ENDC}    { UsingSANE }
014839
014840  {$IFC NOT UsingIncludes}
014841      END.
014842  {$ENDC}
014843
014844
014845  ### END OF FILE SANE.p
014846
```

```
014847
014848  ##########################################################################
014849  ### FILE: Scrap.p
014850  ##########################################################################
014851
014852  {
014853  Created: Sunday, January 6, 1991 at 11:07 PM
014854      Scrap.p
014855      Pascal Interface to the Macintosh Libraries
014856
014857          Copyright Apple Computer, Inc.    1985-1989
014858          All rights reserved
014859  }
014860
014861
014862  {$IFC UNDEFINED UsingIncludes}
014863  {$SETC UsingIncludes := 0}
014864  {$ENDC}
014865
014866  {$IFC NOT UsingIncludes}
014867      UNIT Scrap;
014868      INTERFACE
014869  {$ENDC}
014870
014871  {$IFC UNDEFINED UsingScrap}
014872  {$SETC UsingScrap := 1}
014873
014874  {$I+}
014875  {$SETC ScrapIncludes := UsingIncludes}
014876  {$SETC UsingIncludes := 1}
014877  {$IFC UNDEFINED UsingTypes}
014878  {$I $$Shell(PInterfaces)Types.p}
014879  {$ENDC}
014880  {$SETC UsingIncludes := ScrapIncludes}
014881
014882  TYPE
014883  PScrapStuff = ^ScrapStuff;
014884  ScrapStuff = RECORD
014885      scrapSize: LONGINT;
014886      scrapHandle: Handle;
014887      scrapCount: INTEGER;
014888      scrapState: INTEGER;
014889      scrapName: StringPtr;
014890      END;
014891
014892
014893  FUNCTION InfoScrap: PScrapStuff;
014894      INLINE $A9F9;
014895  FUNCTION UnloadScrap: LONGINT;
014896      INLINE $A9FA;
014897  FUNCTION LoadScrap: LONGINT;
014898      INLINE $A9FB;
014899  FUNCTION GetScrap(hDest: Handle;theType: ResType;VAR offset: LONGINT): LONGINT;
014900      INLINE $A9FD;
014901  FUNCTION ZeroScrap: LONGINT;
014902      INLINE $A9FC;
```

```
014903  FUNCTION PutScrap(length: LONGINT;theType: ResType;source: Ptr): LONGINT;
014904      INLINE $A9FE;
014905
014906
014907  {$ENDC}    { UsingScrap }
014908
014909  {$IFC NOT UsingIncludes}
014910      END.
014911  {$ENDC}
014912
014913
014914  ### END OF FILE Scrap.p
014915
```

```
014916
014917  ############################################################################
014918  ### FILE: Script.p
014919  ############################################################################
014920
014921
014922  {
014923  Created: Tuesday, January 15, 1991 at 8:56 AM
014924   Script.p
014925   Pascal Interface to the Macintosh Libraries
014926
014927    Copyright Apple Computer, Inc. 1986-1991
014928    All rights reserved
014929  }
014930
014931
014932  {$IFC UNDEFINED UsingIncludes}
014933  {$SETC UsingIncludes := 0}
014934  {$ENDC}
014935
014936  {$IFC NOT UsingIncludes}
014937   UNIT Script;
014938   INTERFACE
014939  {$ENDC}
014940
014941  {$IFC UNDEFINED UsingScript}
014942  {$SETC UsingScript := 1}
014943
014944  {$I+}
014945  {$SETC ScriptIncludes := UsingIncludes}
014946  {$SETC UsingIncludes := 1}
014947  {$IFC UNDEFINED UsingTypes}
014948  {$I $$Shell(PInterfaces)Types.p}
014949  {$ENDC}
014950  {$IFC UNDEFINED UsingOSUtils}
014951  {$I $$Shell(PInterfaces)OSUtils.p}
014952  {$ENDC}
014953  {$IFC UNDEFINED UsingQuickdraw}
014954  {$I $$Shell(PInterfaces)Quickdraw.p}
014955  {$ENDC}
014956  {$SETC UsingIncludes := ScriptIncludes}
014957
014958  CONST
014959
014960  { Script System constants }
014961  smSystemScript = -1;                          {designates system script.}
014962  smCurrentScript = -2;                         {designates current font script.}
014963  smAllScripts = -3;                            {designates any script.}
014964  smRoman = 0;                                  {Roman}
014965  smJapanese = 1;                               {Japanese}
014966  smTradChinese = 2;                            {Traditional Chinese}
014967  smKorean = 3;                                 {Korean}
014968  smArabic = 4;                                 {Arabic}
014969  smHebrew = 5;                                 {Hebrew}
014970  smGreek = 6;                                  {Greek}
014971  smCyrillic = 7;                               {Cyrillic}
```

```
014972  smRSymbol = 8;                                  {Right-left symbol}
014973  smDevanagari = 9;                               {Devanagari}
014974  smGurmukhi = 10;                                {Gurmukhi}
014975  smGujarati = 11;                                {Gujarati}
014976  smOriya = 12;                                   {Oriya}
014977  smBengali = 13;                                 {Bengali}
014978  smTamil = 14;                                   {Tamil}
014979  smTelugu = 15;                                  {Telugu}
014980  smKannada = 16;                                 {Kannada/Kanarese}
014981  smMalayalam = 17;                               {Malayalam}
014982  smSinhalese = 18;                               {Sinhalese}
014983  smBurmese = 19;                                 {Burmese}
014984  smKhmer = 20;                                   {Khmer/Cambodian}
014985  smThai = 21;                                    {Thai}
014986  smLaotian = 22;                                 {Laotian}
014987  smGeorgian = 23;                                {Georgian}
014988  smArmenian = 24;                                {Armenian}
014989  smSimpChinese = 25;                             {Simplified Chinese}
014990  smTibetan = 26;                                 {Tibetan}
014991  smMongolian = 27;                               {Mongolian}
014992  smGeez = 28;                                    {Geez/Ethiopic}
014993  smEthiopic = 28;                                {Synonym for smGeez}
014994  smEastEurRoman = 29;                            {Synonym for smSlavic}
014995  smVietnamese = 30;                              {Vietnamese}
014996  smExtArabic = 31;                               {extended Arabic}
014997  smUninterp = 32;                                {uninterpreted symbols, e.g. palette symbols}
014998
014999  {Obsolete names for script systems (kept for backward compatibility)}
015000  smChinese = 2;                                  {(use smTradChinese or smSimpChinese)}
015001  smRussian = 7;                                  {(old name for smCyrillic)}
015002
015003  { smMaldivian = 25;         (no more smMaldivian!)}
015004  smAmharic = 28;                                 {(old name for smGeez)}
015005  smSlavic = 29;                                  {(old name for smEastEurRoman)}
015006  smSindhi = 31;                                  {(old name for smExtArabic)}
015007
015008  { Calendar Codes }
015009  calGregorian = 0;
015010  calArabicCivil = 1;
015011  calArabicLunar = 2;
015012  calJapanese = 3;
015013  calJewish = 4;
015014  calCoptic = 5;
015015  calPersian = 6;
015016
015017  { Integer Format Codes }
015018  intWestern = 0;
015019  intArabic = 1;
015020  intRoman = 2;
015021  intJapanese = 3;
015022  intEuropean = 4;
015023  intOutputMask = $8000;
015024
015025  { CharByte byte types }
015026  smSingleByte = 0;
015027  smFirstByte = -1;
```

```
015028  smLastByte = 1;
015029  smMiddleByte = 2;
015030
015031  { CharType field masks }
015032  smcTypeMask = $000F;
015033  smcReserved = $00F0;
015034  smcClassMask = $0F00;
015035  smcOrientationMask = $1000;                     {two-byte script glyph orientation}
015036  smcRightMask = $2000;
015037  smcUpperMask = $4000;
015038  smcDoubleMask = $8000;
015039
015040  { Basic CharType character types }
015041  smCharPunct = $0000;
015042  smCharAscii = $0001;
015043  smCharEuro = $0007;
015044  smCharExtAscii = $0007;                         { More correct synonym for smCharEuro }
015045
015046  { Additional CharType character types for script systems }
015047  smCharKatakana = $0002;                         {Japanese Katakana}
015048  smCharHiragana = $0003;                         {Japanese Hiragana}
015049  smCharIdeographic = $0004;                      {Hanzi, Kanji, Hanja}
015050  smCharTwoByteGreek = $0005;                     {2-byte Greek in Far East systems}
015051  smCharTwoByteRussian = $0006;                   {2-byte Cyrillic in Far East systems}
015052  smCharBidirect = $0008;                         {Arabic/Hebrew}
015053  smCharHangul = $000C;                           {Korean Hangul}
015054  smCharJamo = $000D;                             {Korean Jamo}
015055
015056  { old names for some of above, for backward compatibility }
015057  smCharFISKana = $0002;                          {Katakana}
015058  smCharFISGana = $0003;                          {Hiragana}
015059  smCharFISIdeo = $0004;                          {Hanzi, Kanji, Hanja}
015060  smCharFISGreek = $0005;                         {2-byte Greek in Far East systems}
015061  smCharFISRussian = $0006;                       {2-byte Cyrillic in Far East systems}
015062
015063  { CharType classes for punctuation (smCharPunct) }
015064  smPunctNormal = $0000;
015065  smPunctNumber = $0100;
015066  smPunctSymbol = $0200;
015067  smPunctBlank = $0300;
015068
015069  { Additional CharType classes for punctuation in two-byte systems }
015070  smPunctRepeat = $0400;                          { repeat marker }
015071  smPunctGraphic = $0500;                         { line graphics }
015072
015073  { CharType Katakana and Hiragana classes for two-byte systems }
015074  smKanaSmall = $0100;                            {small kana character}
015075  smKanaHardOK = $0200;                           {can have dakuten}
015076  smKanaSoftOK = $0300;                           {can have dakuten or han-dakuten}
015077
015078  { CharType Ideographic classes for two-byte systems }
015079  smIdeographicLevel1 = $0000;                    {level 1 char}
015080  smIdeographicLevel2 = $0100;                    {level 2 char}
015081  smIdeographicUser = $0200;                      {user char}
015082
015083  { old names for above, for backward compatibility }
```

```
015084  smFISClassLvl1 = $0000;                         {level 1 char}
015085  smFISClassLvl2 = $0100;                         {level 2 char}
015086  smFISClassUser = $0200;                         {user char}
015087
015088  { CharType Jamo classes for Korean systems }
015089  smJamoJaeum = $0000;                            {simple consonant char}
015090  smJamoBogJaeum = $0100;                         {complex consonant char}
015091  smJamoMoeum = $0200;                            {simple vowel char}
015092  smJamoBogMoeum = $0300;                         {complex vowel char}
015093
015094  { CharType glyph orientation for two-byte systems }
015095  smCharHorizontal = $0000;                       { horizontal character form, or for both }
015096  smCharVertical = $1000;                         { vertical character form }
015097
015098  { CharType directions }
015099  smCharLeft = $0000;
015100  smCharRight = $2000;
015101
015102  { CharType case modifers }
015103  smCharLower = $0000;
015104  smCharUpper = $4000;
015105
015106  { CharType character size modifiers (1 or multiple bytes). }
015107  smChar1byte = $0000;
015108  smChar2byte = $8000;
015109
015110  { Char2Pixel directions }
015111  smLeftCaret = 0;                                {Place caret for left block}
015112  smRightCaret = -1;                              {Place caret for right block}
015113  smHilite = 1;                                   {Direction is TESysJust}
015114
015115  { Transliterate target types for Roman }
015116  smTransAscii = 0;                               {convert to ASCII}
015117  smTransNative = 1;                              {convert to font script}
015118  smTransCase = $FE;                              {convert case for all text}
015119  smTransSystem = $FF;                            {convert to system script}
015120
015121  { Transliterate target types for two-byte scripts }
015122  smTransAscii1 = 2;                              {1-byte Roman}
015123  smTransAscii2 = 3;                              {2-byte Roman}
015124  smTransKana1 = 4;                               {1-byte Japanese Katakana}
015125  smTransKana2 = 5;                               {2-byte Japanese Katakana}
015126  smTransGana2 = 7;                               {2-byte Japanese Hiragana (no 1-byte Hiragana)}
015127  smTransHangul2 = 8;                             {2-byte Korean Hangul}
015128  smTransJamo2 = 9;                               {2-byte Korean Jamo}
015129  smTransBopomofo2 = 10;                          {2-byte Chinese Bopomofo}
015130
015131  { Transliterate target modifiers }
015132  smTransLower = $4000;                           {target becomes lowercase}
015133  smTransUpper = $8000;                           {target becomes uppercase}
015134
015135  { Transliterate source mask - general }
015136  smMaskAll = $FFFFFFFF;                          {Convert all text}
015137
015138  { Transliterate source masks }
015139  smMaskAscii = $00000001;                        {2^smTransAscii}
```

```
015140  smMaskNative = $00000002;                       {2^smTransNative}
015141
015142  { Transliterate source masks for two-byte scripts }
015143  smMaskAscii1 = $00000004;                       {2^smTransAscii1}
015144  smMaskAscii2 = $00000008;                       {2^smTransAscii2}
015145  smMaskKana1 = $00000010;                        {2^smTransKana1}
015146  smMaskKana2 = $00000020;                        {2^smTransKana2}
015147  smMaskGana2 = $00000080;                        {2^smTransGana2}
015148  smMaskHangul2 = $00000100;                      {2^smTransHangul2}
015149  smMaskJamo2 = $00000200;                        {2^smTransJamo2}
015150  smMaskBopomofo2 = $00000400;                    {2^smTransBopomofo2}
015151
015152  { Result values from GetEnvirons, SetEnvirons, GetScript and SetScript calls. }
015153  smNotInstalled = 0;                             {routine not available in script}
015154  smBadVerb = -1;                                 {Bad verb passed to a routine}
015155  smBadScript = -2;                               {Bad script code passed to a routine}
015156
015157  { Values for script redraw flag. }
015158  smRedrawChar = 0;                               {Redraw character only}
015159  smRedrawWord = 1;                               {Redraw entire word (2-byte systems)}
015160  smRedrawLine = -1;                              {Redraw entire line (bidirectional systems)}
015161
015162  { GetEnvirons and SetEnvirons verbs }
015163  smVersion = 0;                                  {Script Manager version number}
015164  smMunged = 2;                                   {Globals change count}
015165  smEnabled = 4;                                  {Count of enabled scripts, incl Roman}
015166  smBidirect = 6;                                 {At least one bidirectional script}
015167  smFontForce = 8;                                {Force font flag}
015168  smIntlForce = 10;                               {Force intl flag}
015169  smForced = 12;                                  {Script was forced to system script}
015170  smDefault = 14;                                 {Script was defaulted to Roman script}
015171  smPrint = 16;                                   {Printer action routine}
015172  smSysScript = 18;                               {System script}
015173  smLastScript = 20;                              {Last keyboard script}
015174  smKeyScript = 22;                               {Keyboard script}
015175  smSysRef = 24;                                  {System folder refNum}
015176  smKeyCache = 26;                                {obsolete}
015177  smKeySwap = 28;                                 {Swapping table handle}
015178  smGenFlags = 30;                                {General flags long}
015179  smOverride = 32;                                {Script override flags}
015180  smCharPortion = 34;                             {Ch vs SpExtra proportion}
015181
015182  { New for System 7.0: }
015183  smDoubleByte = 36;                              {Flag for double-byte script installed}
015184  smKCHRCache = 38;                               {Returns pointer to KCHR cache}
015185  smRegionCode = 40;                              {Returns current region code (verXxx)}
015186
015187  { GetScript and SetScript verbs.
015188  Note: Verbs private to script systems are negative, while
015189  those general across script systems are non-negative. }
015190  smScriptVersion = 0;                            {Script software version}
015191  smScriptMunged = 2;                             {Script entry changed count}
015192  smScriptEnabled = 4;                            {Script enabled flag}
015193  smScriptRight = 6;                              {Right to left flag}
015194  smScriptJust = 8;                               {Justification flag}
015195  smScriptRedraw = 10;                            {Word redraw flag}
```

```
015196  smScriptSysFond = 12;                           {Preferred system font}
015197  smScriptAppFond = 14;                           {Preferred Application font}
015198  smScriptBundle = 16;                            {Beginning of itlb verbs}
015199  smScriptNumber = 16;                            {Script itl0 id}
015200  smScriptDate = 18;                              {Script itl1 id}
015201  smScriptSort = 20;                              {Script itl2 id}
015202  smScriptFlags = 22;                             {flags word}
015203  smScriptToken = 24;                             {Script itl4 id}
015204  smScriptEncoding = 26;                          {id of optional itl5, if present}
015205  smScriptLang = 28;                              {Current language for script}
015206  smScriptNumDate = 30;                           {Script Number/Date formats.}
015207  smScriptKeys = 32;                              {Script KCHR id}
015208  smScriptIcon = 34;                              {ID # of SICN or kcs#/kcs4/kcs8 suite}
015209  smScriptPrint = 36;                             {Script printer action routine}
015210  smScriptTrap = 38;                              {Trap entry pointer}
015211  smScriptCreator = 40;                           {Script file creator}
015212  smScriptFile = 42;                              {Script file name}
015213  smScriptName = 44;                              {Script name}
015214
015215  { There is a hole here for old Kanji private verbs 46-76
015216
015217   New for System 7.0: }
015218  smScriptMonoFondSize = 78;                      {default monospace FOND (hi) & size (lo)}
015219  smScriptPrefFondSize = 80;                      {preferred FOND (hi) & size (lo)}
015220  smScriptSmallFondSize = 82;                     {default small FOND (hi) & size (lo)}
015221  smScriptSysFondSize = 84;                       {default system FOND (hi) & size (lo)}
015222  smScriptAppFondSize = 86;                       {default app FOND (hi) & size (lo)}
015223  smScriptHelpFondSize = 88;                      {default Help Mgr FOND (hi) & size (lo)}
015224  smScriptValidStyles = 90;                       {mask of valid styles for script}
015225  smScriptAliasStyle = 92;                        {style (set) to use for aliases}
015226
015227  { Negative verbs for KeyScript }
015228  smKeyNextScript = -1;                           { Switch to next available script }
015229  smKeySysScript = -2;                            { Switch to the system script }
015230  smKeySwapScript = -3;                           { Switch to previously-used script }
015231
015232  { New for System 7.0: }
015233  smKeyNextKybd = -4;                             { Switch to next keyboard in current keyscript }
015234  smKeySwapKybd = -5;                             { Switch to previously-used keyboard in current keyscript }
015235
015236  smKeyDisableKybds = -6;                         { Disable keyboards not in system or Roman script }
015237  smKeyEnableKybds = -7;                          { Re-enable keyboards for all enabled scripts }
015238  smKeyToggleInline = -8;                         { Toggle inline input for current keyscript }
015239  smKeyToggleDirection = -9;                      { Toggle default line direction (TESysJust) }
015240  smKeyNextInputMethod = -10;                     { Switch to next input method in current keyscript }
015241  smKeySwapInputMethod = -11;                     { Switch to last-used input method in current keyscript }
015242
015243  smKeyDisableKybdSwitch = -12;                   { Disable switching from the current keyboard }
015244
015245
015246  { Bits in the smScriptFlags word
015247  (bits above 7 are non-static) }
015248  smsfIntellCP = 0;                               {Script has intelligent cut & paste}
015249  smsfSingByte = 1;                               {Script has only single bytes}
015250  smsfNatCase = 2;                                {Native chars have upper & lower case}
015251  smsfContext = 3;                                {Script is contextual}
```

```
015252  smsfNoForceFont = 4;                            {Script will not force characters}
015253  smsfB0Digits = 5;                               {Script has alternate digits at B0-B9}
015254  smsfAutoInit = 6;                               {Auto initialize the script}
015255  smsfForms = 13;                                 {Uses contextual forms for letters}
015256  smsfLigatures = 14;                             {Uses contextual ligatures}
015257  smsfReverse = 15;                               {Reverses native text, right-left}
015258
015259  { Bits in the smGenFlags long.
015260  First (high-order) byte is set from itlc flags byte. }
015261  smfShowIcon = 31;                               {Show icon even if only one script}
015262  smfDualCaret = 30;                              {Use dual caret for mixed direction text}
015263  smfNameTagEnab = 29;                            {Reserved for internal use}
015264
015265  { Roman script constants
015266
015267   The following are here for backward compatibility, but should not be used.
015268   This information should be obtained using GetScript. }
015269  romanSysFond = $3FFF;                           {system font id number}
015270  romanAppFond = 3;                               {application font id number}
015271  romanFlags = $0007;                             {roman settings}
015272
015273  { Script Manager font equates. }
015274  smFondStart = $4000;                            {start from 16K}
015275  smFondEnd = $C000;                              {past end of range at 48K}
015276
015277  { Miscellaneous font equates. }
015278  smUprHalfCharSet = $80;                         {first char code in top half of std char set}
015279
015280  { Character Set Extensions }
015281  diaeresisUprY = $D9;
015282  fraction = $DA;
015283  intlCurrency = $DB;
015284  leftSingGuillemet = $DC;
015285  rightSingGuillemet = $DD;
015286  fiLigature = $DE;
015287  flLigature = $DF;
015288  dblDagger = $E0;
015289  centeredDot = $E1;
015290  baseSingQuote = $E2;
015291  baseDblQuote = $E3;
015292  perThousand = $E4;
015293  circumflexUprA = $E5;
015294  circumflexUprE = $E6;
015295  acuteUprA = $E7;
015296  diaeresisUprE = $E8;
015297  graveUprE = $E9;
015298  acuteUprI = $EA;
015299  circumflexUprI = $EB;
015300  diaeresisUprI = $EC;
015301  graveUprI = $ED;
015302  acuteUprO = $EE;
015303  circumflexUprO = $EF;
015304  appleLogo = $F0;
015305  graveUprO = $F1;
015306  acuteUprU = $F2;
015307  circumflexUprU = $F3;
```

```
015308  graveUprU = $F4;
015309  dotlessLwrI = $F5;
015310  circumflex = $F6;
015311  tilde = $F7;
015312  macron = $F8;
015313  breveMark = $F9;
015314  overDot = $FA;
015315  ringMark = $FB;
015316  cedilla = $FC;
015317  doubleAcute = $FD;
015318  ogonek = $FE;
015319  hachek = $FF;
015320
015321  { String2Date status values }
015322  fatalDateTime = $8000;                  {String2Date and String2Time mask to a fatal error}
015323  longDateFound = 1;                      {String2Date mask to long date found}
015324  leftOverChars = 2;                      {String2Date & Time mask to warn of left over characters}
015325  sepNotIntlSep = 4;                      {String2Date & Time mask to warn of non-standard separators}
015326  fieldOrderNotIntl = 8;                  {String2Date & Time mask to warn of non-standard field order}
015327  extraneousStrings = 16;                 {String2Date & Time mask to warn of unparsable strings in text}
015328  tooManySeps = 32;                       {String2Date & Time mask to warn of too many separators}
015329  sepNotConsistent = 64;                  {String2Date & Time mask to warn of inconsistent separators}
015330  tokenErr = $8100;                       {String2Date & Time mask for 'tokenizer err encountered'}
015331  cantReadUtilities = $8200;
015332  dateTimeNotFound = $8400;
015333  dateTimeInvalid = $8800;
015334
015335  { TokenType values }
015336  tokenIntl = 4;                          {the itl resource number of the tokenizer}
015337  tokenEmpty = -1;                        {used internally as an empty flag}
015338  tokenUnknown = 0;                       {chars that do not match a defined token type}
015339  tokenWhite = 1;                         {white space}
015340  tokenLeftLit = 2;                       {literal begin}
015341  tokenRightLit = 3;                      {literal end}
015342  tokenAlpha = 4;                         {alphabetic}
015343  tokenNumeric = 5;                       {numeric}
015344  tokenNewLine = 6;                       {new line}
015345  tokenLeftComment = 7;                   {open comment}
015346  tokenRightComment = 8;                  {close comment}
015347  tokenLiteral = 9;                       {literal}
015348  tokenEscape = 10;                       {character escape (e.g. '\' in "\n", "\t")}
015349  tokenAltNum = 11;                       {alternate number (e.g. $B0-B9 in Arabic,Hebrew)}
015350  tokenRealNum = 12;                      {real number}
015351  tokenAltReal = 13;                      {alternate real number}
015352  tokenReserve1 = 14;                     {reserved}
015353  tokenReserve2 = 15;                     {reserved}
015354  tokenLeftParen = 16;                    {open parenthesis}
015355  tokenRightParen = 17;                   {close parenthesis}
015356  tokenLeftBracket = 18;                  {open square bracket}
015357  tokenRightBracket = 19;                 {close square bracket}
015358  tokenLeftCurly = 20;                    {open curly bracket}
015359  tokenRightCurly = 21;                   {close curly bracket}
015360  tokenLeftEnclose = 22;                  {open guillemet}
015361  tokenRightEnclose = 23;                 {close guillemet}
015362  tokenPlus = 24;
015363  tokenMinus = 25;
```

```
015364  tokenAsterisk = 26;                     {times/multiply}
015365  tokenDivide = 27;
015366  tokenPlusMinus = 28;                    {plus or minus symbol}
015367  tokenSlash = 29;
015368  tokenBackSlash = 30;
015369  tokenLess = 31;                         {less than symbol}
015370  tokenGreat = 32;                        {greater than symbol}
015371  tokenEqual = 33;
015372  tokenLessEqual2 = 34;                   {less than or equal, 2 characters (e.g. <=)}
015373  tokenLessEqual1 = 35;                   {less than or equal, 1 character}
015374  tokenGreatEqual2 = 36;                  {greater than or equal, 2 characters (e.g. >=)}
015375  tokenGreatEqual1 = 37;                  {greater than or equal, 1 character}
015376  token2Equal = 38;                       {double equal (e.g. ==)}
015377  tokenColonEqual = 39;                   {colon equal}
015378  tokenNotEqual = 40;                     {not equal, 1 character}
015379  tokenLessGreat = 41;                    {less/greater, Pascal not equal (e.g. <>)}
015380  tokenExclamEqual = 42;                  {exclamation equal, C not equal (e.g. !=)}
015381  tokenExclam = 43;                       {exclamation point}
015382  tokenTilde = 44;                        {centered tilde}
015383  tokenComma = 45;
015384  tokenPeriod = 46;
015385  tokenLeft2Quote = 47;                   {open double quote}
015386  tokenRight2Quote = 48;                  {close double quote}
015387  tokenLeft1Quote = 49;                   {open single quote}
015388  tokenRight1Quote = 50;                  {close single quote}
015389  token2Quote = 51;                       {double quote}
015390  token1Quote = 52;                       {single quote}
015391  tokenSemicolon = 53;
015392  tokenPercent = 54;
015393  tokenCaret = 55;
015394  tokenUnderline = 56;
015395  tokenAmpersand = 57;
015396  tokenAtSign = 58;
015397  tokenBar = 59;                          {vertical bar}
015398  tokenQuestion = 60;
015399  tokenPi = 61;                           {lower-case pi}
015400  tokenRoot = 62;                         {square root symbol}
015401  tokenSigma = 63;                        {capital sigma}
015402  tokenIntegral = 64;                     {integral sign}
015403  tokenMicro = 65;
015404  tokenCapPi = 66;                        {capital pi}
015405  tokenInfinity = 67;
015406  tokenColon = 68;
015407  tokenHash = 69;                         {e.g. #}
015408  tokenDollar = 70;
015409  tokenNoBreakSpace = 71;                 {non-breaking space}
015410  tokenFraction = 72;
015411  tokenIntlCurrency = 73;
015412  tokenLeftSingGuillemet = 74;
015413  tokenRightSingGuillemet = 75;
015414  tokenPerThousand = 76;
015415  tokenEllipsis = 77;
015416  tokenCenterDot = 78;
015417  tokenNil = 127;
015418  delimPad = -2;
015419
```

```
015420  { obsolete, misspelled token names kept for backward compatibility }
015421  tokenTilda = 44;
015422  tokenCarat = 55;
015423
015424  { the NumberParts indices }
015425  tokLeftQuote = 1;
015426  tokRightQuote = 2;
015427  tokLeadPlacer = 3;
015428  tokLeader = 4;
015429  tokNonLeader = 5;
015430  tokZeroLead = 6;
015431  tokPercent = 7;
015432  tokPlusSign = 8;
015433  tokMinusSign = 9;
015434  tokThousands = 10;
015435  tokSeparator = 12;                          {11 is a reserved field}
015436  tokEscape = 13;
015437  tokDecPoint = 14;
015438  tokEPlus = 15;
015439  tokEMinus = 16;
015440  tokMaxSymbols = 31;
015441
015442  curNumberPartsVersion = 1;                  {current version of NumberParts record}
015443  fVNumber = 0;                               {first version of NumFormatString}
015444
015445  { Date equates }
015446  smallDateBit = 31;                          {Restrict valid date/time to range of Time global}
015447  togChar12HourBit = 30;                      {If toggling hour by char, accept hours 1..12 only}
015448  togCharZCycleBit = 29;                      {Modifier for togChar12HourBit: accept hours 0..11 only}
015449  togDelta12HourBit = 28;                     {If toggling hour up/down, restrict to 12-hour range (am/pm)}
015450  genCdevRangeBit = 27;                       {Restrict date/time to range used by genl CDEV}
015451  validDateFields = -1;
015452  maxDateField = 10;
015453
015454  eraMask = $0001;
015455  yearMask = $0002;
015456  monthMask = $0004;
015457  dayMask = $0008;
015458  hourMask = $0010;
015459  minuteMask = $0020;
015460  secondMask = $0040;
015461  dayOfWeekMask = $0080;
015462  dayOfYearMask = $0100;
015463  weekOfYearMask = $0200;
015464  pmMask = $0400;
015465  dateStdMask = $007F;                        {default for ValidDate flags and ToggleDate TogglePB.togFlags}
015466
015467  { Toggle results }
015468  toggleUndefined = 0;
015469  toggleOK = 1;
015470  toggleBadField = 2;
015471  toggleBadDelta = 3;
015472  toggleBadChar = 4;
015473  toggleUnknown = 5;
015474  toggleBadNum = 6;
015475  toggleOutOfRange = 7;                       {synonym for toggleErr3}
```

```
015476  toggleErr3 = 7;
015477  toggleErr4 = 8;
015478  toggleErr5 = 9;
015479
015480  { New constants for System 7.0:
015481
015482    Constants for truncWhere argument in TruncString and TruncText }
015483  smTruncEnd = 0;                             { Truncate at end }
015484  smTruncMiddle = $4000;                      { Truncate in middle }
015485
015486  { Constants for TruncString and TruncText results }
015487  smNotTruncated = 0;                         { No truncation was necessary }
015488  smTruncated = 1;                            { Truncation performed }
015489  smTruncErr = -1;                            { General error }
015490
015491  {Constants for styleRunPosition argument in NPortionText, NDrawJust,
015492   NMeasureJust, NChar2Pixel, and NPixel2Char.}
015493  smOnlyStyleRun = 0;                         { This is the only style run on the line }
015494  smLeftStyleRun = 1;                         { This is leftmost of multiple style runs on the line }
015495  smRightStyleRun = 2;                        { This is rightmost of multiple style runs on the line }
015496  smMiddleStyleRun = 3;                       { There are multiple style runs on the line and this
015497   is neither the leftmost nor the rightmost. }
015498
015499  TYPE
015500  TokenResults = (tokenOK,tokenOverflow,stringOverflow,badDelim,badEnding,
015501   crash);
015502
015503  LongDateField = (eraField,yearField,monthField,dayField,hourField,minuteField,
015504   secondField,dayOfWeekField,dayOfYearField,weekOfYearField,pmField,res1Field,
015505   res2Field,res3Field);
015506
015507  StyledLineBreakCode = (smBreakWord,smBreakChar,smBreakOverflow);
015508
015509  FormatClass = (fPositive,fNegative,fZero);
015510
015511  FormatResultType = (fFormatOK,fBestGuess,fOutOfSynch,fSpuriousChars,fMissingDelimiter,
015512   fExtraDecimal,fMissingLiteral,fExtraExp,fFormatOverflow,fFormStrIsNAN,
015513   fBadPartsTable,fExtraPercent,fExtraSeparator,fEmptyFormatString);
015514
015515
015516  CharByteTable = PACKED ARRAY [0..255] OF SignedByte;
015517  ToggleResults = INTEGER;
015518
015519  BreakTablePtr = ^BreakTable;
015520  BreakTable = RECORD
015521   charTypes: ARRAY [0..255] OF SignedByte;
015522   tripleLength: INTEGER;
015523   triples: ARRAY [0..0] OF INTEGER;
015524   END;
015525
015526  { New NBreakTable for System 7.0: }
015527  NBreakTablePtr = ^NBreakTable;
015528  NBreakTable = RECORD
015529   flags1: SignedByte;
015530   flags2: SignedByte;
015531   version: INTEGER;
```

```
015532   classTableOff: INTEGER;
015533   auxCTableOff: INTEGER;
015534   backwdTableOff: INTEGER;
015535   forwdTableOff: INTEGER;
015536   doBackup: INTEGER;
015537   reserved: INTEGER;
015538   charTypes: ARRAY [0..255] OF SignedByte;
015539   tables: ARRAY [0..0] OF INTEGER;
015540   END;
015541
015542  OffPair = RECORD
015543   offFirst: INTEGER;
015544   offSecond: INTEGER;
015545   END;
015546
015547
015548  OffsetTable = ARRAY [0..2] OF OffPair;
015549
015550  ItlcRecord = RECORD
015551   itlcSystem: INTEGER;                          {default system script}
015552   itlcReserved: INTEGER;                        {reserved}
015553   itlcFontForce: SignedByte;                    {default font force flag}
015554   itlcIntlForce: SignedByte;                    {default intl force flag}
015555   itlcOldKybd: SignedByte;                      {MacPlus intl keybd flag}
015556   itlcFlags: SignedByte;                        {general flags}
015557   itlcIconOffset: INTEGER;                      {keyboard icon offset; not used in 7.0}
015558   itlcIconSide: SignedByte;                     {keyboard icon side; not used in 7.0}
015559   itlcIconRsvd: SignedByte;                     {rsvd for other icon info}
015560   itlcRegionCode: INTEGER;                      {preferred verXxx code}
015561   itlcReserved3: ARRAY [0..33] OF SignedByte;   {for future use}
015562   END;
015563
015564  ItlbRecord = RECORD
015565   itlbNumber: INTEGER;                          {itl0 id number}
015566   itlbDate: INTEGER;                            {itl1 id number}
015567   itlbSort: INTEGER;                            {itl2 id number}
015568   itlbFlags: INTEGER;                           {Script flags}
015569   itlbToken: INTEGER;                           {itl4 id number}
015570   itlbEncoding: INTEGER;                        {itl5 ID # (optional; char encoding)}
015571   itlbLang: INTEGER;                            {current language for script }
015572   itlbNumRep: SignedByte;                       {number representation code}
015573   itlbDateRep: SignedByte;                      {date representation code }
015574   itlbKeys: INTEGER;                            {KCHR id number}
015575   itlbIcon: INTEGER;                            {ID # of SICN or kcs#/kcs4/kcs8 suite.}
015576   END;
015577
015578  { New ItlbExtRecord structure for System 7.0 }
015579  ItlbExtRecord = RECORD
015580   base: ItlbRecord;                             {un-extended ItlbRecord}
015581   itlbLocalSize: LONGINT;                       {size of script's local record}
015582   itlbMonoFond: INTEGER;                        {default monospace FOND ID}
015583   itlbMonoSize: INTEGER;                        {default monospace font size}
015584   itlbPrefFond: INTEGER;                        {preferred FOND ID}
015585   itlbPrefSize: INTEGER;                        {preferred font size}
015586   itlbSmallFond: INTEGER;                       {default small FOND ID}
015587   itlbSmallSize: INTEGER;                       {default small font size}
```

```
015588   itlbSysFond: INTEGER;                          {default system FOND ID}
015589   itlbSysSize: INTEGER;                          {default system font size}
015590   itlbAppFond: INTEGER;                          {default application FOND ID}
015591   itlbAppSize: INTEGER;                          {default application font size}
015592   itlbHelpFond: INTEGER;                         {default Help Mgr FOND ID}
015593   itlbHelpSize: INTEGER;                         {default Help Mgr font size}
015594   itlbValidStyles: Style;                        {set of valid styles for script}
015595   itlbAliasStyle: Style;                         {style (set) to mark aliases}
015596   END;
015597
015598  MachineLocation = RECORD
015599   latitude: Fract;
015600   longitude: Fract;
015601   CASE INTEGER OF
015602     0:
015603    (dlsDelta: SignedByte);                       {signed byte; daylight savings delta}
015604     1:
015605    (gmtDelta: LONGINT);                          {must mask - see documentation}
015606   END;
015607
015608
015609  String2DateStatus = INTEGER;
015610  TokenType = INTEGER;
015611  DelimType = ARRAY [0..1] OF TokenType;
015612  CommentType = ARRAY [0..3] OF TokenType;
015613
015614  TokenRecPtr = ^TokenRec;
015615  TokenRec = RECORD
015616   theToken: TokenType;
015617   position: Ptr;                                 {pointer into original source}
015618   length: LONGINT;                               {length of text in original source}
015619   stringPosition: StringPtr;                     {Pascal/C string copy of identifier}
015620   END;
015621
015622  TokenBlockPtr = ^TokenBlock;
015623  TokenBlock = RECORD
015624   source: Ptr;                                   {pointer to stream of characters}
015625   sourceLength: LONGINT;                         {length of source stream}
015626   tokenList: Ptr;                                {pointer to array of tokens}
015627   tokenLength: LONGINT;                          {maximum length of TokenList}
015628   tokenCount: LONGINT;                           {number tokens generated by tokenizer}
015629   stringList: Ptr;                               {pointer to stream of identifiers}
015630   stringLength: LONGINT;                         {length of string list}
015631   stringCount: LONGINT;                          {number of bytes currently used}
015632   doString: BOOLEAN;                             {make strings & put into StringList}
015633   doAppend: BOOLEAN;                             {append to TokenList rather than replace}
015634   doAlphanumeric: BOOLEAN;                       {identifiers may include numeric}
015635   doNest: BOOLEAN;                               {do comments nest?}
015636   leftDelims: ARRAY [0..1] OF TokenType;
015637   rightDelims: ARRAY [0..1] OF TokenType;
015638   leftComment: ARRAY [0..3] OF TokenType;
015639   rightComment: ARRAY [0..3] OF TokenType;
015640   escapeCode: TokenType;                         {escape symbol code}
015641   decimalCode: TokenType;
015642   itlResource: Handle;                           {handle to itl4 resource of current script}
015643   reserved: ARRAY [0..7] OF LONGINT;             {must be zero!}
```

```
015644   END;
015645
015646  UntokenTablePtr = ^UntokenTable;
015647  UntokenTableHandle = ^UntokenTablePtr;
015648  UntokenTable = RECORD
015649   len: INTEGER;
015650   lastToken: INTEGER;
015651   index: ARRAY [0..255] OF INTEGER;              {index table; last = lastToken}
015652   END;
015653
015654  DateCachePtr = ^DateCacheRecord;
015655  DateCacheRecord = PACKED RECORD
015656   hidden: ARRAY [0..255] OF INTEGER;             {only for temporary use}
015657   END;
015658
015659
015660  LongDateTime = comp;
015661
015662  LongDateCvt = RECORD
015663   CASE INTEGER OF
015664     0:
015665    (c: Comp);
015666     1:
015667    (lHigh: LONGINT;
015668    lLow: LONGINT);
015669   END;
015670
015671  LongDateRec = RECORD
015672   CASE INTEGER OF
015673     0:
015674    (era: INTEGER;
015675    year: INTEGER;
015676    month: INTEGER;
015677    day: INTEGER;
015678    hour: INTEGER;
015679    minute: INTEGER;
015680    second: INTEGER;
015681    dayOfWeek: INTEGER;
015682    dayOfYear: INTEGER;
015683    weekOfYear: INTEGER;
015684    pm: INTEGER;
015685    res1: INTEGER;
015686    res2: INTEGER;
015687    res3: INTEGER);
015688     1:
015689    (list: ARRAY [0..13] OF INTEGER);              {Index by LongDateField!}
015690     2:
015691    (eraAlt: INTEGER;
015692    oldDate: DateTimeRec);
015693   END;
015694
015695
015696  DateDelta = SignedByte;
015697
015698  TogglePB = RECORD
015699   togFlags: LONGINT;                             {caller normally sets low word to dateStdMask=$7F}
```

```
015700   amChars: ResType;                              {from 'itl0', but uppercased}
015701   pmChars: ResType;                              {from 'itl0', but uppercased}
015702   reserved: ARRAY [0..3] OF LONGINT;
015703   END;
015704
015705
015706  FormatOrder = ARRAY [0..0] OF INTEGER;
015707  FormatOrderPtr = ^FormatOrder;
015708  FormatStatus = INTEGER;
015709
015710  WideChar = RECORD
015711   CASE BOOLEAN OF
015712     TRUE:
015713    (a: PACKED ARRAY [0..1] OF CHAR);             {0 is the high order character}
015714     FALSE:
015715    (b: INTEGER);
015716   END;
015717
015718  WideCharArr = RECORD
015719   size: INTEGER;
015720   data: PACKED ARRAY [0..9] OF WideChar;
015721   END;
015722
015723  NumFormatString = PACKED RECORD
015724   fLength: Byte;
015725   fVersion: Byte;
015726   data: PACKED ARRAY [0..253] OF SignedByte;     {private data}
015727   END;
015728
015729  Itl4Ptr = ^Itl4Rec;
015730  Itl4Handle = ^Itl4Ptr;
015731  Itl4Rec = RECORD
015732   flags: INTEGER;                                {reserved}
015733   resourceType: LONGINT;                         {contains 'itl4'}
015734   resourceNum: INTEGER;                          {resource ID}
015735   version: INTEGER;                              {version number}
015736   resHeader1: LONGINT;                           {reserved}
015737   resHeader2: LONGINT;                           {reserved}
015738   numTables: INTEGER;                            {number of tables, one-based}
015739   mapOffset: LONGINT;                            {offset to table that maps byte to token}
015740   strOffset: LONGINT;                            {offset to routine that copies canonical string}
015741   fetchOffset: LONGINT;                          {offset to routine that gets next byte of character}
015742   unTokenOffset: LONGINT;                        {offset to table that maps token to canonical string}
015743   defPartsOffset: LONGINT;                       {offset to default number parts table}
015744   resOffset6: LONGINT;                           {reserved}
015745   resOffset7: LONGINT;                           {reserved}
015746   resOffset8: LONGINT;                           {reserved}
015747   END;
015748
015749  { New NItl4Rec for System 7.0: }
015750  NItl4Ptr = ^NItl4Rec;
015751  NItl4Handle = ^NItl4Ptr;
015752  NItl4Rec = RECORD
015753   flags: INTEGER;                                {reserved}
015754   resourceType: LONGINT;                         {contains 'itl4'}
015755   resourceNum: INTEGER;                          {resource ID}
```

```
015756   version: INTEGER;                              {version number}
015757   format: INTEGER;                               {format code}
015758   resHeader: INTEGER;                            {reserved}
015759   resHeader2: LONGINT;                           {reserved}
015760   numTables: INTEGER;                            {number of tables, one-based}
015761   mapOffset: LONGINT;                            {offset to table that maps byte to token}
015762   strOffset: LONGINT;                            {offset to routine that copies canonical string}
015763   fetchOffset: LONGINT;                          {offset to routine that gets next byte of character}
015764   unTokenOffset: LONGINT;                        {offset to table that maps token to canonical string}
015765   defPartsOffset: LONGINT;                       {offset to default number parts table}
015766   whtSpListOffset: LONGINT;                      {offset to white space code list}
015767   resOffset7: LONGINT;                           {reserved}
015768   resOffset8: LONGINT;                           {reserved}
015769   resLength1: INTEGER;                           {reserved}
015770   resLength2: INTEGER;                           {reserved}
015771   resLength3: INTEGER;                           {reserved}
015772   unTokenLength: INTEGER;                        {length of untoken table}
015773   defPartsLength: INTEGER;                       {length of default number parts table}
015774   whtSpListLength: INTEGER;                      {length of white space code list}
015775   resLength7: INTEGER;                           {reserved}
015776   resLength8: INTEGER;                           {reserved}
015777   END;
015778
015779  NumberPartsPtr = ^NumberParts;
015780  NumberParts = RECORD
015781   version: INTEGER;
015782   data: ARRAY [1..31] OF WideChar;               {index by [tokLeftQuote..tokMaxSymbols]}
015783   pePlus: WideCharArr;
015784   peMinus: WideCharArr;
015785   peMinusPlus: WideCharArr;
015786   altNumTable: WideCharArr;
015787   reserved: PACKED ARRAY [0..19] OF CHAR;
015788   END;
015789
015790  FVector = RECORD
015791   start: INTEGER;
015792   length: INTEGER;
015793   END;
015794
015795
015796  TripleInt = ARRAY [0..2] OF FVector;            { index by [fPositive..fZero] }
015797
015798  ScriptRunStatus = RECORD
015799   script: SignedByte;
015800   variant: SignedByte;
015801   END;
015802
015803
015804  { New types for System 7.0:
015805
015806    type for truncWhere parameter in new TruncString, TruncText }
015807  TruncCode = INTEGER;
015808
015809  { type for styleRunPosition parameter in NPixel2Char etc. }
015810  JustStyleCode = INTEGER;
015811
```

```
015812  FUNCTION FontScript: INTEGER;
015813   INLINE $2F3C,$8200,$0000,$A8B5;
015814  FUNCTION IntlScript: INTEGER;
015815   INLINE $2F3C,$8200,$0002,$A8B5;
015816  PROCEDURE KeyScript(code: INTEGER);
015817   INLINE $2F3C,$8002,$0004,$A8B5;
015818  FUNCTION Font2Script(fontNumber: INTEGER): INTEGER;
015819   INLINE $2F3C,$8202,$0006,$A8B5;
015820  FUNCTION GetEnvirons(verb: INTEGER): LONGINT;
015821   INLINE $2F3C,$8402,$0008,$A8B5;
015822  FUNCTION SetEnvirons(verb: INTEGER;param: LONGINT): OSErr;
015823   INLINE $2F3C,$8206,$000A,$A8B5;
015824  FUNCTION GetScript(script: INTEGER;verb: INTEGER): LONGINT;
015825   INLINE $2F3C,$8404,$000C,$A8B5;
015826  FUNCTION SetScript(script: INTEGER;verb: INTEGER;param: LONGINT): OSErr;
015827   INLINE $2F3C,$8208,$000E,$A8B5;
015828  FUNCTION CharByte(textBuf: Ptr;textOffset: INTEGER): INTEGER;
015829   INLINE $2F3C,$8206,$0010,$A8B5;
015830  FUNCTION CharType(textBuf: Ptr;textOffset: INTEGER): INTEGER;
015831   INLINE $2F3C,$8206,$0012,$A8B5;
015832  FUNCTION Pixel2Char(textBuf: Ptr;textLen: INTEGER;slop: INTEGER;pixelWidth: INTEGER;
015833   VAR leadingEdge: BOOLEAN): INTEGER;
015834   INLINE $2F3C,$820E,$0014,$A8B5;
015835  FUNCTION Char2Pixel(textBuf: Ptr;textLen: INTEGER;slop: INTEGER;offset: INTEGER;
015836   direction: INTEGER): INTEGER;
015837   INLINE $2F3C,$820C,$0016,$A8B5;
015838  FUNCTION Transliterate(srcHandle: Handle;dstHandle: Handle;target: INTEGER;
015839   srcMask: LONGINT): OSErr;
015840   INLINE $2F3C,$820E,$0018,$A8B5;
015841  PROCEDURE FindWord(textPtr: Ptr;textLength: INTEGER;offset: INTEGER;leadingEdge: BOOLEAN;
015842   breaks: BreakTablePtr;VAR offsets: OffsetTable);
015843   INLINE $2F3C,$8012,$001A,$A8B5;
015844  PROCEDURE HiliteText(textPtr: Ptr;textLength: INTEGER;firstOffset: INTEGER;
015845   secondOffset: INTEGER;VAR offsets: OffsetTable);
015846   INLINE $2F3C,$800E,$001C,$A8B5;
015847  PROCEDURE DrawJust(textPtr: Ptr;textLength: INTEGER;slop: INTEGER);
015848   INLINE $2F3C,$8008,$001E,$A8B5;
015849  PROCEDURE MeasureJust(textPtr: Ptr;textLength: INTEGER;slop: INTEGER;charLocs: Ptr);
015850   INLINE $2F3C,$800C,$0020,$A8B5;
015851  FUNCTION ParseTable(VAR table: CharByteTable): BOOLEAN;
015852   INLINE $2F3C,$8204,$0022,$A8B5;
015853  FUNCTION GetDefFontSize: INTEGER;
015854   INLINE $3EB8,$0BA8,$6604,$3EBC,$000C;
015855  FUNCTION GetSysFont: INTEGER;
015856   INLINE $3EB8,$0BA6;
015857  FUNCTION GetAppFont: INTEGER;
015858   INLINE $3EB8,$0984;
015859  FUNCTION GetMBarHeight: INTEGER;
015860   INLINE $3EB8,$0BAA;
015861  FUNCTION GetSysJust: INTEGER;
015862   INLINE $3EB8,$0BAC;
015863  PROCEDURE SetSysJust(newJust: INTEGER);
015864   INLINE $31DF,$0BAC;
015865  PROCEDURE ReadLocation(VAR loc: MachineLocation);
015866   INLINE $205F,$203C,$000C,$00E4,$A051;
015867  PROCEDURE WriteLocation(loc: MachineLocation);
```

```
015868   INLINE $205F,$203C,$000C,$00E4,$A052;
015869  PROCEDURE UprText(textPtr: Ptr;len: INTEGER);
015870   INLINE $301F,$205F,$A054;
015871  PROCEDURE LwrText(textPtr: Ptr;len: INTEGER);
015872   INLINE $301F,$205F,$A056;
015873
015874
015875  {  New for 7.0  }
015876  PROCEDURE LowerText(textPtr: Ptr;len: INTEGER);
015877   INLINE $301F,$205F,$A056;
015878  PROCEDURE StripText(textPtr: Ptr;len: INTEGER);
015879   INLINE $301F,$205F,$A256;
015880  PROCEDURE UpperText(textPtr: Ptr;len: INTEGER);
015881   INLINE $301F,$205F,$A456;
015882  PROCEDURE StripUpperText(textPtr: Ptr;len: INTEGER);
015883   INLINE $301F,$205F,$A656;
015884
015885  FUNCTION StyledLineBreak(textPtr: Ptr;textLen: LONGINT;textStart: LONGINT;
015886   textEnd: LONGINT;flags: LONGINT;VAR textWidth: Fixed;VAR textOffset: LONGINT): StyledLineBreakCode;
015887   INLINE $2F3C,$821C,$FFFE,$A8B5;
015888  PROCEDURE GetFormatOrder(ordering: FormatOrderPtr;firstFormat: INTEGER;
015889   lastFormat: INTEGER;lineRight: BOOLEAN;rlDirProc: Ptr;dirParam: Ptr);
015890   INLINE $2F3C,$8012,$FFFC,$A8B5;
015891  FUNCTION IntlTokenize(tokenParam: TokenBlockPtr): TokenResults;
015892   INLINE $2F3C,$8204,$FFFA,$A8B5;
015893  FUNCTION InitDateCache(theCache: DateCachePtr): OSErr;
015894   INLINE $2F3C,$8204,$FFF8,$A8B5;
015895  FUNCTION String2Date(textPtr: Ptr;textLen: LONGINT;theCache: DateCachePtr;
015896   VAR lengthUsed: LONGINT;VAR dateTime: LongDateRec): String2DateStatus;
015897   INLINE $2F3C,$8214,$FFF6,$A8B5;
015898  FUNCTION String2Time(textPtr: Ptr;textLen: LONGINT;theCache: DateCachePtr;
015899   VAR lengthUsed: LONGINT;VAR dateTime: LongDateRec): String2DateStatus;
015900   INLINE $2F3C,$8214,$FFF4,$A8B5;
015901  PROCEDURE LongDate2Secs(lDate: LongDateRec;VAR lSecs: LongDateTime);
015902   INLINE $2F3C,$8008,$FFF2,$A8B5;
015903  PROCEDURE LongSecs2Date(VAR lSecs: LongDateTime;VAR lDate: LongDateRec);
015904   INLINE $2F3C,$8008,$FFF0,$A8B5;
015905  FUNCTION ToggleDate(VAR lSecs: LongDateTime;field: LongDateField;delta: DateDelta;
015906   ch: INTEGER;params: TogglePB): ToggleResults;
015907   INLINE $2F3C,$820E,$FFEE,$A8B5;
015908  FUNCTION Str2Format(inString: Str255;partsTable: NumberParts;VAR outString: NumFormatString): FormatStatus;
015909   INLINE $2F3C,$820C,$FFEC,$A8B5;
015910  FUNCTION Format2Str(myCanonical: NumFormatString;partsTable: NumberParts;
015911   VAR outString: Str255;VAR positions: TripleInt): FormatStatus;
015912   INLINE $2F3C,$8210,$FFEA,$A8B5;
015913  FUNCTION FormatX2Str(x: Extended80;myCanonical: NumFormatString;partsTable: NumberParts;
015914   VAR outString: Str255): FormatStatus;
015915   INLINE $2F3C,$8210,$FFE8,$A8B5;
015916  FUNCTION FormatStr2X(source: Str255;myCanonical: NumFormatString;partsTable: NumberParts;
015917   VAR x: Extended80): FormatStatus;
015918   INLINE $2F3C,$8210,$FFE6,$A8B5;
015919  FUNCTION PortionText(textPtr: Ptr;textLen: LONGINT): Fixed;
015920   INLINE $2F3C,$8408,$0024,$A8B5;
015921  FUNCTION FindScriptRun(textPtr: Ptr;textLen: LONGINT;VAR lenUsed: LONGINT): ScriptRunStatus;
015922   INLINE $2F3C,$820C,$0026,$A8B5;
015923  FUNCTION VisibleLength(textPtr: Ptr;textLen: LONGINT): LONGINT;
```

```
015924   INLINE $2F3C,$8408,$0028,$A8B5;
015925  FUNCTION ValidDate(vDate: LongDateRec;flags: LONGINT;VAR newSecs: LongDateTime): INTEGER;
015926   INLINE $2F3C,$820C,$FFE4,$A8B5;
015927
015928
015929  {  New for 7.0  }
015930  PROCEDURE NFindWord(textPtr: Ptr;textLength: INTEGER;offset: INTEGER;leadingEdge: BOOLEAN;
015931   nbreaks: NBreakTablePtr;VAR offsets: OffsetTable);
015932   INLINE $2F3C,$8012,$FFE2,$A8B5;
015933  FUNCTION TruncString(width: INTEGER;VAR theString: Str255;truncWhere: TruncCode): INTEGER;
015934   INLINE $2F3C,$8208,$FFE0,$A8B5;
015935  FUNCTION TruncText(width: INTEGER;textPtr: Ptr;VAR length: INTEGER;truncWhere: TruncCode): INTEGER;
015936   INLINE $2F3C,$820C,$FFDE,$A8B5;
015937  FUNCTION ReplaceText(baseText: Handle;substitutionText: Handle;key: Str15): INTEGER;
015938   INLINE $2F3C,$820C,$FFDC,$A8B5;
015939  FUNCTION NPixel2Char(textBuf: Ptr;textLen: LONGINT;slop: Fixed;pixelWidth: Fixed;
015940   VAR leadingEdge: BOOLEAN;VAR widthRemaining: Fixed;styleRunPosition: JustStyleCode;
015941   numer: Point;denom: Point): INTEGER;
015942   INLINE $2F3C,$8222,$002E,$A8B5;
015943  FUNCTION NChar2Pixel(textBuf: Ptr;textLen: LONGINT;slop: Fixed;offset: LONGINT;
015944   direction: INTEGER;styleRunPosition: JustStyleCode;numer: Point;denom: Point): INTEGER;
015945   INLINE $2F3C,$821C,$0030,$A8B5;
015946  PROCEDURE NDrawJust(textPtr: Ptr;textLength: LONGINT;slop: Fixed;styleRunPosition: JustStyleCode;
015947   numer: Point;denom: Point);
015948   INLINE $2F3C,$8016,$0032,$A8B5;
015949  PROCEDURE NMeasureJust(textPtr: Ptr;textLength: LONGINT;slop: Fixed;charLocs: Ptr;
015950   styleRunPosition: JustStyleCode;numer: Point;denom: Point);
015951   INLINE $2F3C,$801A,$0034,$A8B5;
015952  FUNCTION NPortionText(textPtr: Ptr;textLen: LONGINT;styleRunPosition: JustStyleCode;
015953   numer: Point;denom: Point): Fixed;
015954   INLINE $2F3C,$8412,$0036,$A8B5;
015955
015956
015957  {$ENDC} { UsingScript }
015958
015959  {$IFC NOT UsingIncludes}
015960   END.
015961  {$ENDC}
015962
015963
015964  ### END OF FILE Script.p
015965
```

```
015966
015967  ############################################################################
015968  ### FILE: SCSI.p
015969  ############################################################################
015970
015971  {
015972  Created: Sunday, January 6, 1991 at 11:14 PM
015973      SCSI.p
015974      Pascal Interface to the Macintosh Libraries
015975
015976          Copyright Apple Computer, Inc.    1986-1990
015977          All rights reserved
015978  }
015979
015980
015981  {$IFC UNDEFINED UsingIncludes}
015982  {$SETC UsingIncludes := 0}
015983  {$ENDC}
015984
015985  {$IFC NOT UsingIncludes}
015986      UNIT SCSI;
015987      INTERFACE
015988  {$ENDC}
015989
015990  {$IFC UNDEFINED UsingSCSI}
015991  {$SETC UsingSCSI := 1}
015992
015993  {$I+}
015994  {$SETC SCSIIncludes := UsingIncludes}
015995  {$SETC UsingIncludes := 1}
015996  {$IFC UNDEFINED UsingTypes}
015997  {$I $$Shell(PInterfaces)Types.p}
015998  {$ENDC}
015999  {$SETC UsingIncludes := SCSIIncludes}
016000
016001  CONST
016002  scInc = 1;
016003  scNoInc = 2;
016004  scAdd = 3;
016005  scMove = 4;
016006  scLoop = 5;
016007  scNop = 6;
016008  scStop = 7;
016009  scComp = 8;
016010  scCommErr = 2;                              {communications error, operation timeout}
016011  scArbNBErr = 3;                             {arbitration timeout waiting for not BSY}
016012  scBadParmsErr = 4;                          {bad parameter or TIB opcode}
016013  scPhaseErr = 5;                             {SCSI bus not in correct phase for attempted operation}
016014  scCompareErr = 6;                           {data compare error}
016015  scMgrBusyErr = 7;                           {SCSI Manager busy }
016016  scSequenceErr = 8;                          {attempted operation is out of sequence}
016017  scBusTOErr = 9;                             {CPU bus timeout}
016018  scComplPhaseErr = 10;                       {SCSI bus wasn't in Status phase}
016019  sbSIGWord = $4552;
016020  pMapSIG = $504D;
016021
```

```
016022  TYPE
016023  Block0 = PACKED RECORD
016024      sbSig: INTEGER;                         {unique value for SCSI block 0}
016025      sbBlkSize: INTEGER;                     {block size of device}
016026      sbBlkCount: LONGINT;                    {number of blocks on device}
016027      sbDevType: INTEGER;                     {device type}
016028      sbDevId: INTEGER;                       {device id}
016029      sbData: LONGINT;                        {not used}
016030      sbDrvrCount: INTEGER;                   {driver descriptor count}
016031      ddBlock: LONGINT;                       {1st driver's starting block}
016032      ddSize: INTEGER;                        {size of 1st driver (512-byte blks)}
016033      ddType: INTEGER;                        {system type (1 for Mac+)}
016034      ddPad: ARRAY [0..242] OF INTEGER;       {ARRAY[0..242] OF INTEGER; not used}
016035      END;
016036
016037  Partition = PACKED RECORD
016038      pmSig: INTEGER;                         {unique value for map entry blk}
016039      pmSigPad: INTEGER;                      {currently unused}
016040      pmMapBlkCnt: LONGINT;                   {# of blks in partition map}
016041      pmPyPartStart: LONGINT;                 {physical start blk of partition}
016042      pmPartBlkCnt: LONGINT;                  {# of blks in this partition}
016043      pmPartName: PACKED ARRAY [0..31] OF CHAR;   {ASCII partition name}
016044      pmParType: PACKED ARRAY [0..31] OF CHAR;    {ASCII partition type}
016045      pmLgDataStart: LONGINT;                 {log. # of partition's 1st data blk}
016046      pmDataCnt: LONGINT;                     {# of blks in partition's data area}
016047      pmPartStatus: LONGINT;                  {bit field for partition status}
016048      pmLgBootStart: LONGINT;                 {log. blk of partition's boot code}
016049      pmBootSize: LONGINT;                    {number of bytes in boot code}
016050      pmBootAddr: LONGINT;                    {memory load address of boot code}
016051      pmBootAddr2: LONGINT;                   {currently unused}
016052      pmBootEntry: LONGINT;                   {entry point of boot code}
016053      pmBootEntry2: LONGINT;                  {currently unused}
016054      pmBootCksum: LONGINT;                   {checksum of boot code}
016055      pmProcessor: PACKED ARRAY [0..15] OF CHAR;  {ASCII for the processor type}
016056      pmPad: ARRAY [0..187] OF INTEGER;       {512 bytes long currently unused}
016057      END;
016058
016059  SCSIInstr = RECORD
016060      scOpcode: INTEGER;
016061      scParam1: LONGINT;
016062      scParam2: LONGINT;
016063      END;
016064
016065
016066  FUNCTION SCSIReset: OSErr;
016067      INLINE $4267,$A815;
016068  FUNCTION SCSIGet: OSErr;
016069      INLINE $3F3C,$0001,$A815;
016070  FUNCTION SCSISelect(targetID: INTEGER): OSErr;
016071      INLINE $3F3C,$0002,$A815;
016072  FUNCTION SCSICmd(buffer: Ptr;count: INTEGER): OSErr;
016073      INLINE $3F3C,$0003,$A815;
016074  FUNCTION SCSIRead(tibPtr: Ptr): OSErr;
016075      INLINE $3F3C,$0005,$A815;
016076  FUNCTION SCSIRBlind(tibPtr: Ptr): OSErr;
016077      INLINE $3F3C,$0008,$A815;
```

```
016078  FUNCTION SCSIWrite(tibPtr: Ptr): OSErr;
016079      INLINE $3F3C,$0006,$A815;
016080  FUNCTION SCSIWBlind(tibPtr: Ptr): OSErr;
016081      INLINE $3F3C,$0009,$A815;
016082  FUNCTION SCSIComplete(VAR stat: INTEGER;VAR message: INTEGER;wait: LONGINT): OSErr;
016083      INLINE $3F3C,$0004,$A815;
016084  FUNCTION SCSIStat: INTEGER;
016085      INLINE $3F3C,$000A,$A815;
016086  FUNCTION SCSISelAtn(targetID: INTEGER): OSErr;
016087      INLINE $3F3C,$000B,$A815;
016088  FUNCTION SCSIMsgIn(VAR message: INTEGER): OSErr;
016089      INLINE $3F3C,$000C,$A815;
016090  FUNCTION SCSIMsgOut(message: INTEGER): OSErr;
016091      INLINE $3F3C,$000D,$A815;
016092
016093
016094  {$ENDC}    { UsingSCSI }
016095
016096  {$IFC NOT UsingIncludes}
016097      END.
016098  {$ENDC}
016099
016100
016101  ### END OF FILE SCSI.p
016102
```

```
016103
016104  ************************************************************************
016105  *** FILE: SCSIIntf.p
016106  ************************************************************************
016107
016108  {
016109      File: SCSIIntf.p
016110
016111      As of MPW 3.0, interface files were reorganized to more closely
016112      match "Inside Macintosh" reference books and be more consistant
016113      from language to language.
016114
016115      Interfaces for the SCSI Manager are now found in SCSI.p.
016116      This file, which includes SCSI.p, is provided for compatibility
016117      with old sources.
016118
016119      Pascal Interface to the Macintosh Libraries
016120      Copyright Apple Computer, Inc. 1988
016121      All Rights Reserved
016122  }
016123
016124  {$IFC UNDEFINED UsingIncludes}
016125  {$SETC UsingIncludes := 0}
016126  {$ENDC}
016127
016128  {$IFC NOT UsingIncludes}
016129      UNIT SCSIIntf;
016130      INTERFACE
016131  {$ENDC}
016132
016133  {$IFC UNDEFINED UsingSCSIIntf}
016134  {$SETC UsingSCSIIntf := 1}
016135
016136  {$I+}
016137  {$SETC SCSIIntfIncludes := UsingIncludes}
016138  {$SETC UsingIncludes := 1}
016139  {$IFC UNDEFINED UsingSCSI}
016140  {$I $$Shell(PInterfaces)SCSI.p}
016141  {$ENDC}
016142  {$SETC UsingIncludes := SCSIIntfIncludes}
016143
016144  {$ENDC}    { UsingSCSIIntf }
016145
016146  {$IFC NOT UsingIncludes}
016147      END.
016148  {$ENDC}
016149
016150
016151  ### END OF FILE SCSIIntf.p
016152
```

```
016153
016154  ############################################################################
016155  ### FILE: SegLoad.p
016156  ############################################################################
016157
016158  {
016159  Created: Sunday, January 6, 1991 at 11:15 PM
016160      SegLoad.p
016161      Pascal Interface to the Macintosh Libraries
016162
016163          Copyright Apple Computer, Inc.    1985-1990
016164          All rights reserved
016165  }
016166
016167
016168  {$IFC UNDEFINED UsingIncludes}
016169  {$SETC UsingIncludes := 0}
016170  {$ENDC}
016171
016172  {$IFC NOT UsingIncludes}
016173      UNIT SegLoad;
016174      INTERFACE
016175  {$ENDC}
016176
016177  {$IFC UNDEFINED UsingSegLoad}
016178  {$SETC UsingSegLoad := 1}
016179
016180  {$I+}
016181  {$SETC SegLoadIncludes := UsingIncludes}
016182  {$SETC UsingIncludes := 1}
016183  {$IFC UNDEFINED UsingTypes}
016184  {$I $$Shell(PInterfaces)Types.p}
016185  {$ENDC}
016186  {$SETC UsingIncludes := SegLoadIncludes}
016187
016188  CONST
016189  appOpen = 0;            {Open the Document (s)}
016190  appPrint = 1;           {Print the Document (s)}
016191
016192  TYPE
016193  AppFile = RECORD
016194      vRefNum: INTEGER;
016195      fType: OSType;
016196      versNum: INTEGER;   {versNum in high byte}
016197      fName: Str255;
016198      END;
016199
016200
016201  PROCEDURE UnloadSeg(routineAddr: Ptr);
016202      INLINE $A9F1;
016203  PROCEDURE ExitToShell;
016204      INLINE $A9F4;
016205  PROCEDURE GetAppParms(VAR apName: Str255;VAR apRefNum: INTEGER;VAR apParam: Handle);
016206      INLINE $A9F5;
016207  PROCEDURE CountAppFiles(VAR message: INTEGER;VAR count: INTEGER);
016208  PROCEDURE GetAppFiles(index: INTEGER;VAR theFile: AppFile);
```

```
016209  PROCEDURE ClrAppFiles(index: INTEGER);
016210
016211
016212  {$ENDC}    { UsingSegLoad }
016213
016214  {$IFC NOT UsingIncludes}
016215      END.
016216  {$ENDC}
016217
016218
016219  ### END OF FILE SegLoad.p
016220
```

```
016221
016222  ############################################################################
016223  ### FILE: Serial.p
016224  ############################################################################
016225
016226  {
016227  Created: Sunday, January 6, 1991 at 11:15 PM
016228      Serial.p
016229      Pascal Interface to the Macintosh Libraries
016230
016231          Copyright Apple Computer, Inc.    1985-1990
016232          All rights reserved
016233  }
016234
016235
016236  {$IFC UNDEFINED UsingIncludes}
016237  {$SETC UsingIncludes := 0}
016238  {$ENDC}
016239
016240  {$IFC NOT UsingIncludes}
016241      UNIT Serial;
016242      INTERFACE
016243  {$ENDC}
016244
016245  {$IFC UNDEFINED UsingSerial}
016246  {$SETC UsingSerial := 1}
016247
016248  {$I+}
016249  {$SETC SerialIncludes := UsingIncludes}
016250  {$SETC UsingIncludes := 1}
016251  {$IFC UNDEFINED UsingTypes}
016252  {$I $$Shell(PInterfaces)Types.p}
016253  {$ENDC}
016254  {$SETC UsingIncludes := SerialIncludes}
016255
016256  CONST
016257  baud300 = 380;
016258  baud600 = 189;
016259  baud1200 = 94;
016260  baud1800 = 62;
016261  baud2400 = 46;
016262  baud3600 = 30;
016263  baud4800 = 22;
016264  baud7200 = 14;
016265  baud9600 = 10;
016266  baud19200 = 4;
016267  baud57600 = 0;
016268  stop10 = 16384;
016269  stop15 = -32768;
016270  stop20 = -16384;
016271  noParity = 0;
016272  oddParity = 4096;
016273  evenParity = 12288;
016274  data5 = 0;
016275  data6 = 2048;
016276  data7 = 1024;
```

```
016277  data8 = 3072;
016278  ctsEvent = 32;
016279  breakEvent = 128;
016280  xOffWasSent = 128;
016281  dtrNegated = 64;
016282  ainRefNum = -6;     {serial port A input}
016283  aoutRefNum = -7;    {serial port A output}
016284  binRefNum = -8;     {serial port B input}
016285  boutRefNum = -9;    {serial port B output}
016286
016287  TYPE
016288  SPortSel = (sPortA,sPortB);
016289
016290
016291  SerShk = PACKED RECORD
016292      fXOn: Byte;     {XOn flow control enabled flag}
016293      fCTS: Byte;     {CTS flow control enabled flag}
016294      xOn: CHAR;      {XOn character}
016295      xOff: CHAR;     {XOff character}
016296      errs: Byte;     {errors mask bits}
016297      evts: Byte;     {event enable mask bits}
016298      fInX: Byte;     {Input flow control enabled flag}
016299      fDTR: Byte;     {DTR input flow control flag}
016300      END;
016301
016302  SerStaRec = PACKED RECORD
016303      cumErrs: Byte;
016304      xOffSent: Byte;
016305      rdPend: Byte;
016306      wrPend: Byte;
016307      ctsHold: Byte;
016308      xOffHold: Byte;
016309      END;
016310
016311
016312  FUNCTION SerReset(refNum: INTEGER;serConfig: INTEGER): OSErr;
016313  FUNCTION SerSetBuf(refNum: INTEGER;serBPtr: Ptr;serBLen: INTEGER): OSErr;
016314  FUNCTION SerHShake(refNum: INTEGER;flags: SerShk): OSErr;
016315  FUNCTION SerSetBrk(refNum: INTEGER): OSErr;
016316  FUNCTION SerClrBrk(refNum: INTEGER): OSErr;
016317  FUNCTION SerGetBuf(refNum: INTEGER;VAR count: LONGINT): OSErr;
016318  FUNCTION SerStatus(refNum: INTEGER;VAR serSta: SerStaRec): OSErr;
016319
016320
016321  {$ENDC}    { UsingSerial }
016322
016323  {$IFC NOT UsingIncludes}
016324      END.
016325  {$ENDC}
016326
016327
016328  ### END OF FILE Serial.p
016329
```

```
016330
016331  ############################################################################
016332  ### FILE: ShutDown.p
016333  ############################################################################
016334
016335  {
016336  Created: Sunday, January 6, 1991 at 11:15 PM
016337      ShutDown.p
016338      Pascal Interface to the Macintosh Libraries
016339
016340          Copyright Apple Computer, Inc.    1987-1989
016341          All rights reserved
016342  }
016343
016344
016345  {$IFC UNDEFINED UsingIncludes}
016346  {$SETC UsingIncludes := 0}
016347  {$ENDC}
016348
016349  {$IFC NOT UsingIncludes}
016350      UNIT ShutDown;
016351      INTERFACE
016352  {$ENDC}
016353
016354  {$IFC UNDEFINED UsingShutDown}
016355  {$SETC UsingShutDown := 1}
016356
016357  {$I+}
016358  {$SETC ShutDownIncludes := UsingIncludes}
016359  {$SETC UsingIncludes := 1}
016360  {$IFC UNDEFINED UsingTypes}
016361  {$I $$Shell(PInterfaces)Types.p}
016362  {$ENDC}
016363  {$SETC UsingIncludes := ShutDownIncludes}
016364
016365  CONST
016366  sdOnPowerOff = 1;       {call procedure before power off.}
016367  sdOnRestart = 2;        {call procedure before restart.}
016368  sdOnUnmount = 4;        {call procedure before unmounting.}
016369  sdOnDrivers = 8;        {call procedure before closing drivers.}
016370  sdRestartOrPower = 3;   {call before either power off or restart.}
016371
016372  PROCEDURE ShutDwnPower;
016373      INLINE $3F3C,$0001,$A895;
016374  PROCEDURE ShutDwnStart;
016375      INLINE $3F3C,$0002,$A895;
016376  PROCEDURE ShutDwnInstall(shutDownProc: ProcPtr;flags: INTEGER);
016377      INLINE $3F3C,$0003,$A895;
016378  PROCEDURE ShutDwnRemove(shutDownProc: ProcPtr);
016379      INLINE $3F3C,$0004,$A895;
016380
016381
016382  {$ENDC}    { UsingShutDown }
016383
016384  {$IFC NOT UsingIncludes}
016385      END.
```

```
016386  {$ENDC}
016387
016388
016389  ### END OF FILE ShutDown.p
016390
```

```
016391
016392  ##########################################################################
016393  ### FILE: Signal.p
016394  ##########################################################################
016395
016396
016397  {
016398  Created: Friday, August 2, 1991 at 11:40 PM
016399   Signal.p
016400   Pascal Interface to the Macintosh Libraries
016401
016402      Signal Handling interface.
016403      This must be compatible with C's <signal.h>
016404
016405    Copyright Apple Computer, Inc. 1986, 1987, 1988, 1991
016406    All rights reserved
016407  }
016408
016409
016410  {$IFC UNDEFINED UsingIncludes}
016411  {$SETC UsingIncludes := 0}
016412  {$ENDC}
016413
016414  {$IFC NOT UsingIncludes}
016415   UNIT Signal;
016416   INTERFACE
016417  {$ENDC}
016418
016419  {$IFC UNDEFINED UsingSignal}
016420  {$SETC UsingSignal := 1}
016421
016422
016423  TYPE
016424      SignalMap =     INTEGER;
016425      SignalHandler = ^LONGINT;   { Pointer to function }
016426
016427  CONST
016428      SIG_ERR =      -1;          { Returned by IEsignal on error }
016429      SIG_IGN =       0;
016430      SIG_DFL =       1;
016431      SIG_HOLD =      3;
016432      SIG_RELEASE =   5;
016433
016434      SIGABRT =       $0001;
016435      SIGINT =        $0002;      { Currently only SIGINT implemented }
016436      SIGFPE =        $0004;
016437      SIGILL =        $0008;
016438      SIGSEGV =       $0010;
016439      SIGTERM =       $0020;
016440
016441  { Signal Handling Functions }
016442
016443  FUNCTION
016444      IEsignal(sigNum: LONGINT; sigHdlr: UNIV SignalHandler):
016445      SignalHandler; C;
016446
```

```
016447  FUNCTION
016448      IEraise(sigNum: LONGINT):
016449      LONGINT; C;
016450
016451
016452  {$ENDC} { UsingSignal }
016453
016454  {$IFC NOT UsingIncludes}
016455   END.
016456  {$ENDC}
016457
016458  ### END OF FILE Signal.p
016459
```

```
016460
016461  ############################################################################
016462  ### FILE: Slots.p
016463  ############################################################################
016464
016465  {
016466  Created: Sunday, January 6, 1991 at 11:16 PM
016467      Slots.p
016468      Pascal Interface to the Macintosh Libraries
016469
016470          Copyright Apple Computer, Inc.    1986-1990
016471          All rights reserved
016472  }
016473
016474
016475  {$IFC UNDEFINED UsingIncludes}
016476  {$SETC UsingIncludes := 0}
016477  {$ENDC}
016478
016479  {$IFC NOT UsingIncludes}
016480      UNIT Slots;
016481      INTERFACE
016482  {$ENDC}
016483
016484  {$IFC UNDEFINED UsingSlots}
016485  {$SETC UsingSlots := 1}
016486
016487  {$I+}
016488  {$SETC SlotsIncludes := UsingIncludes}
016489  {$SETC UsingIncludes := 1}
016490  {$IFC UNDEFINED UsingTypes}
016491  {$I $$Shell(PInterfaces)Types.p}
016492  {$ENDC}
016493  {$IFC UNDEFINED UsingOSEvents}
016494  {$I $$Shell(PInterfaces)OSEvents.p}
016495  {$ENDC}
016496  {$IFC UNDEFINED UsingOSUtils}
016497  {$I $$Shell(PInterfaces)OSUtils.p}
016498  {$ENDC}
016499  {$IFC UNDEFINED UsingFiles}
016500  {$I $$Shell(PInterfaces)Files.p}
016501  {$ENDC}
016502  {$SETC UsingIncludes := SlotsIncludes}
016503
016504  CONST
016505  fCardIsChanged = 1;                             {Card is Changed field in StatusFlags field of sInfoArray}
016506  fCkForSame = 0;                                 {For SearchSRT. Flag to check for SAME sResource in the table. }
016507  fCkForNext = 1;                                 {For SearchSRT. Flag to check for NEXT sResource in the table. }
016508  fWarmStart = 2;                                 {If this bit is set then warm start else cold start.}
016509
016510  stateNil = 0;                                   {State}
016511  stateSDMInit = 1;                               {:Slot declaration manager Init}
016512  statePRAMInit = 2;                              {:sPRAM record init}
016513  statePInit = 3;                                 {:Primary init}
016514  stateSInit = 4;                                 {:Secondary init}
016515
```

```
016516  { flags for spParamData }
016517  fall = 0;                                       { bit 0: set=search enabled/disabled sRsrc's }
016518  foneslot = 1;                                   {       1: set=search sRsrc's in given slot only }
016519  fnext = 2;                                      {       2: set=search for next sRsrc }
016520
016521  TYPE
016522  SQElemPtr = ^SlotIntQElement;
016523  SlotIntQElement = RECORD
016524      sqLink: Ptr;                                {ptr to next element}
016525      sqType: INTEGER;                            {queue type ID for validity}
016526      sqPrio: INTEGER;                            {priority}
016527      sqAddr: ProcPtr;                            {interrupt service routine}
016528      sqParm: LONGINT;                            {optional A1 parameter}
016529      END;
016530
016531  SpBlockPtr = ^SpBlock;
016532  SpBlock = PACKED RECORD
016533      spResult: LONGINT;                          {FUNCTION Result}
016534      spsPointer: Ptr;                            {structure pointer}
016535      spSize: LONGINT;                            {size of structure}
016536      spOffsetData: LONGINT;                      {offset/data field used by sOffsetData}
016537      spIOFileName: Ptr;                          {ptr to IOFile name for sDisDrvrName}
016538      spsExecPBlk: Ptr;                           {pointer to sExec parameter block.}
016539      spParamData: LONGINT;                       {misc parameter data (formerly spStackPtr).}
016540      spMisc: LONGINT;                            {misc field for SDM.}
016541      spReserved: LONGINT;                        {reserved for future expansion}
016542      spIOReserved: INTEGER;                      {Reserved field of Slot Resource Table}
016543      spRefNum: INTEGER;                          {RefNum}
016544      spCategory: INTEGER;                        {sType: Category}
016545      spCType: INTEGER;                           {Type}
016546      spDrvrSW: INTEGER;                          {DrvrSW}
016547      spDrvrHW: INTEGER;                          {DrvrHW}
016548      spTBMask: SignedByte;                       {type bit mask bits 0..3 mask words 0..3}
016549      spSlot: SignedByte;                         {slot number}
016550      spID: SignedByte;                           {structure ID}
016551      spExtDev: SignedByte;                       {ID of the external device}
016552      spHwDev: SignedByte;                        {Id of the hardware device.}
016553      spByteLanes: SignedByte;                    {bytelanes from card ROM format block}
016554      spFlags: SignedByte;                        {standard flags}
016555      spKey: SignedByte;                          {Internal use only}
016556      END;
016557
016558  SInfoRecPtr = ^SInfoRecord;
016559  SInfoRecord = PACKED RECORD
016560      siDirPtr: Ptr;                              {Pointer to directory}
016561      siInitStatusA: INTEGER;                     {initialization E}
016562      siInitStatusV: INTEGER;                     {status returned by vendor init code}
016563      siState: SignedByte;                        {initialization state}
016564      siCPUByteLanes: SignedByte;                 {0=[d0..d7] 1=[d8..d15]}
016565      siTopOfROM: SignedByte;                     {Top of ROM= $FssFFFFx: x is TopOfROM}
016566      siStatusFlags: SignedByte;                  {bit 0 - card is changed}
016567      siTOConst: INTEGER;                         {Time Out C for BusErr}
016568      siReserved: PACKED ARRAY [0..1] OF SignedByte;  {reserved}
016569      siROMAddr: Ptr;                             { addr of top of ROM }
016570      siSlot: CHAR;                               { slot number }
016571      siPadding: PACKED ARRAY [0..2] OF SignedByte;   { reserved }
```

```
016572      END;
016573
016574  SDMRecord = PACKED RECORD
016575      sdBEVSave: ProcPtr;                         {Save old BusErr vector}
016576      sdBusErrProc: ProcPtr;                      {Go here to determine if it is a BusErr}
016577      sdErrorEntry: ProcPtr;                      {Go here if BusErrProc finds real BusErr}
016578      sdReserved: LONGINT;                        {Reserved}
016579      END;
016580
016581  FHeaderRecPtr = ^FHeaderRec;
016582  FHeaderRec = PACKED RECORD
016583      fhDirOffset: LONGINT;                       {offset to directory}
016584      fhLength: LONGINT;                          {length of ROM}
016585      fhCRC: LONGINT;                             {CRC}
016586      fhROMRev: SignedByte;                       {revision of ROM}
016587      fhFormat: SignedByte;                       {format - 2}
016588      fhTstPat: LONGINT;                          {test pattern}
016589      fhReserved: SignedByte;                     {reserved}
016590      fhByteLanes: SignedByte;                    {ByteLanes}
016591      END;
016592
016593  SEBlock = PACKED RECORD
016594      seSlot: SignedByte;                         {Slot number.}
016595      sesRsrcId: SignedByte;                      {sResource Id.}
016596      seStatus: INTEGER;                          {Status of code executed by sExec.}
016597      seFlags: SignedByte;                        {Flags}
016598      seFiller0: SignedByte;                      {Filler, must be SignedByte to align on odd boundry}
016599      seFiller1: SignedByte;                      {Filler}
016600      seFiller2: SignedByte;                      {Filler}
016601      seResult: LONGINT;                          {Result of sLoad.}
016602      seIOFileName: LONGINT;                      {Pointer to IOFile name.}
016603      seDevice: SignedByte;                       {Which device to read from.}
016604      sePartition: SignedByte;                    {The partition.}
016605      seOSType: SignedByte;                       {Type of OS.}
016606      seReserved: SignedByte;                     {Reserved field.}
016607      seRefNum: SignedByte;                       {RefNum of the driver.}
016608      seNumDevices: SignedByte;                   { Number of devices to load.}
016609      seBootState: SignedByte;                    {State of StartBoot code.}
016610      END;
016611
016612
016613
016614  {  Principle  }
016615  FUNCTION SReadByte(spBlkPtr: SpBlockPtr): OSErr;
016616      INLINE $205F,$7000,$A06E,$3E80;
016617  FUNCTION SReadWord(spBlkPtr: SpBlockPtr): OSErr;
016618      INLINE $205F,$7001,$A06E,$3E80;
016619  FUNCTION SReadLong(spBlkPtr: SpBlockPtr): OSErr;
016620      INLINE $205F,$7002,$A06E,$3E80;
016621  FUNCTION SGetCString(spBlkPtr: SpBlockPtr): OSErr;
016622      INLINE $205F,$7003,$A06E,$3E80;
016623  FUNCTION SGetBlock(spBlkPtr: SpBlockPtr): OSErr;
016624      INLINE $205F,$7005,$A06E,$3E80;
016625  FUNCTION SFindStruct(spBlkPtr: SpBlockPtr): OSErr;
016626      INLINE $205F,$7006,$A06E,$3E80;
016627  FUNCTION SReadStruct(spBlkPtr: SpBlockPtr): OSErr;
```

```
016628      INLINE $205F,$7007,$A06E,$3E80;
016629
016630
016631  {  Special  }
016632  FUNCTION SReadInfo(spBlkPtr: SpBlockPtr): OSErr;
016633      INLINE $205F,$7010,$A06E,$3E80;
016634  FUNCTION SReadPRAMRec(spBlkPtr: SpBlockPtr): OSErr;
016635      INLINE $205F,$7011,$A06E,$3E80;
016636  FUNCTION SPutPRAMRec(spBlkPtr: SpBlockPtr): OSErr;
016637      INLINE $205F,$7012,$A06E,$3E80;
016638  FUNCTION SReadFHeader(spBlkPtr: SpBlockPtr): OSErr;
016639      INLINE $205F,$7013,$A06E,$3E80;
016640  FUNCTION SNextSRsrc(spBlkPtr: SpBlockPtr): OSErr;
016641      INLINE $205F,$7014,$A06E,$3E80;
016642  FUNCTION SNextTypeSRsrc(spBlkPtr: SpBlockPtr): OSErr;
016643      INLINE $205F,$7015,$A06E,$3E80;
016644  FUNCTION SRsrcInfo(spBlkPtr: SpBlockPtr): OSErr;
016645      INLINE $205F,$7016,$A06E,$3E80;
016646  FUNCTION SCkCardStat(spBlkPtr: SpBlockPtr): OSErr;
016647      INLINE $205F,$7018,$A06E,$3E80;
016648  FUNCTION SReadDrvrName(spBlkPtr: SpBlockPtr): OSErr;
016649      INLINE $205F,$7019,$A06E,$3E80;
016650  FUNCTION SFindDevBase(spBlkPtr: SpBlockPtr): OSErr;
016651      INLINE $205F,$701B,$A06E,$3E80;
016652  FUNCTION SFindBigDevBase(spBlkPtr: SpBlockPtr): OSErr;
016653      INLINE $205F,$701C,$A06E,$3E80;
016654
016655
016656  {  Advanced  }
016657  FUNCTION InitSDeclMgr(spBlkPtr: SpBlockPtr): OSErr;
016658      INLINE $205F,$7020,$A06E,$3E80;
016659  FUNCTION SPrimaryInit(spBlkPtr: SpBlockPtr): OSErr;
016660      INLINE $205F,$7021,$A06E,$3E80;
016661  FUNCTION SCardChanged(spBlkPtr: SpBlockPtr): OSErr;
016662      INLINE $205F,$7022,$A06E,$3E80;
016663  FUNCTION SExec(spBlkPtr: SpBlockPtr): OSErr;
016664      INLINE $205F,$7023,$A06E,$3E80;
016665  FUNCTION SOffsetData(spBlkPtr: SpBlockPtr): OSErr;
016666      INLINE $205F,$7024,$A06E,$3E80;
016667  FUNCTION SInitPRAMRecs(spBlkPtr: SpBlockPtr): OSErr;
016668      INLINE $205F,$7025,$A06E,$3E80;
016669  FUNCTION SReadPBSize(spBlkPtr: SpBlockPtr): OSErr;
016670      INLINE $205F,$7026,$A06E,$3E80;
016671  FUNCTION SCalcStep(spBlkPtr: SpBlockPtr): OSErr;
016672      INLINE $205F,$7028,$A06E,$3E80;
016673  FUNCTION SInitSRsrcTable(spBlkPtr: SpBlockPtr): OSErr;
016674      INLINE $205F,$7029,$A06E,$3E80;
016675  FUNCTION SSearchSRT(spBlkPtr: SpBlockPtr): OSErr;
016676      INLINE $205F,$702A,$A06E,$3E80;
016677  FUNCTION SUpdateSRT(spBlkPtr: SpBlockPtr): OSErr;
016678      INLINE $205F,$702B,$A06E,$3E80;
016679  FUNCTION SCalcSPointer(spBlkPtr: SpBlockPtr): OSErr;
016680      INLINE $205F,$702C,$A06E,$3E80;
016681  FUNCTION SGetDriver(spBlkPtr: SpBlockPtr): OSErr;
016682      INLINE $205F,$702D,$A06E,$3E80;
016683  FUNCTION SPtrToSlot(spBlkPtr: SpBlockPtr): OSErr;
```

```
016684      INLINE $205F,$702E,$A06E,$3E80;
016685  FUNCTION SFindSInfoRecPtr(spBlkPtr: SpBlockPtr): OSErr;
016686      INLINE $205F,$702F,$A06E,$3E80;
016687  FUNCTION SFindSRsrcPtr(spBlkPtr: SpBlockPtr): OSErr;
016688      INLINE $205F,$7030,$A06E,$3E80;
016689  FUNCTION SDeleteSRTRec(spBlkPtr: SpBlockPtr): OSErr;
016690      INLINE $205F,$7031,$A06E,$3E80;
016691
016692  FUNCTION OpenSlot(paramBlock: ParmBlkPtr;async: BOOLEAN): OSErr;
016693  FUNCTION OpenSlotSync(paramBlock: ParmBlkPtr): OSErr;
016694      INLINE $205F,$A200,$3E80;
016695  FUNCTION OpenSlotAsync(paramBlock: ParmBlkPtr): OSErr;
016696      INLINE $205F,$A600,$3E80;
016697
016698
016699  {  Device Manager Slot Support  }
016700  FUNCTION SIntInstall(sIntQElemPtr: SQElemPtr;theSlot: INTEGER): OSErr;
016701      INLINE $301F,$205F,$A075,$3E80;
016702  FUNCTION SIntRemove(sIntQElemPtr: SQElemPtr;theSlot: INTEGER): OSErr;
016703      INLINE $301F,$205F,$A076,$3E80;
016704
016705  FUNCTION SVersion(spBlkPtr: SpBlockPtr): OSErr;
016706      INLINE $205F,$7008,$A06E,$3E80;
016707  FUNCTION SetSRsrcState(spBlkPtr: SpBlockPtr): OSErr;
016708      INLINE $205F,$7009,$A06E,$3E80;
016709  FUNCTION InsertSRTRec(spBlkPtr: SpBlockPtr): OSErr;
016710      INLINE $205F,$700A,$A06E,$3E80;
016711  FUNCTION SGetSRsrc(spBlkPtr: SpBlockPtr): OSErr;
016712      INLINE $205F,$700B,$A06E,$3E80;
016713  FUNCTION SGetTypeSRsrc(spBlkPtr: SpBlockPtr): OSErr;
016714      INLINE $205F,$700C,$A06E,$3E80;
016715  FUNCTION SGetSRsrcPtr(spBlkPtr: SpBlockPtr): OSErr;
016716      INLINE $205F,$701D,$A06E,$3E80;
016717
016718
016719  {$ENDC}    { UsingSlots }
016720
016721  {$IFC NOT UsingIncludes}
016722      END.
016723  {$ENDC}
016724
016725
016726  ### END OF FILE Slots.p
016727
```

```
016728
016729  ##########################################################################
016730  ### FILE: Sound.p
016731  ##########################################################################
016732
016733
016734  {
016735  Created: Monday, December 2, 1991 at 5:09 PM
016736   Sound.p
016737   Pascal Interface to the Macintosh Libraries
016738
016739    Copyright Apple Computer, Inc. 1986-1991
016740    All rights reserved
016741  }
016742
016743
016744  {$IFC UNDEFINED UsingIncludes}
016745  {$SETC UsingIncludes := 0}
016746  {$ENDC}
016747
016748  {$IFC NOT UsingIncludes}
016749   UNIT Sound;
016750   INTERFACE
016751  {$ENDC}
016752
016753  {$IFC UNDEFINED UsingSound}
016754  {$SETC UsingSound := 1}
016755
016756  {$I+}
016757  {$SETC SoundIncludes := UsingIncludes}
016758  {$SETC UsingIncludes := 1}
016759  {$IFC UNDEFINED UsingTypes}
016760  {$I $$Shell(PInterfaces)Types.p}
016761  {$ENDC}
016762  {$IFC UNDEFINED UsingFiles}
016763  {$I $$Shell(PInterfaces)Files.p}
016764  {$ENDC}
016765  {$SETC UsingIncludes := SoundIncludes}
016766
016767  CONST
016768  swMode = -1;                              { Sound Driver modes }
016769  ftMode = 1;
016770  ffMode = 0;
016771
016772  synthCodeRsrc = 'snth';                   { Resource types used by Sound Manager }
016773  soundListRsrc = 'snd ';
016774
016775  twelfthRootTwo = 1.05946309434;
016776  rate22khz = $56EE8BA3;                    { 22254.54545 in fixed-point }
016777  rate11khz = $2B7745D1;                    { 11127.27273 in fixed-point }
016778
016779  { synthesizer numbers for SndNewChannel }
016780  squareWaveSynth = 1;                      {square wave synthesizer}
016781  waveTableSynth = 3;                       {wave table synthesizer}
016782  sampledSynth = 5;                         {sampled sound synthesizer}
016783
```

```
016784  { old Sound Manager MACE synthesizer numbers }
016785  MACE3snthID = 11;
016786  MACE6snthID = 13;
016787
016788  { command numbers for SndDoCommand and SndDoImmediate }
016789  nullCmd = 0;
016790  initCmd = 1;
016791  freeCmd = 2;
016792  quietCmd = 3;
016793  flushCmd = 4;
016794  reInitCmd = 5;
016795
016796  waitCmd = 10;
016797  pauseCmd = 11;
016798  resumeCmd = 12;
016799  callBackCmd = 13;
016800  syncCmd = 14;
016801  emptyCmd = 15;
016802
016803  tickleCmd = 20;
016804  requestNextCmd = 21;
016805  howOftenCmd = 22;
016806  wakeUpCmd = 23;
016807  availableCmd = 24;
016808  versionCmd = 25;
016809  totalLoadCmd = 26;
016810  loadCmd = 27;
016811
016812  scaleCmd = 30;
016813  tempoCmd = 31;
016814
016815  freqDurationCmd = 40;
016816  restCmd = 41;
016817  freqCmd = 42;
016818  ampCmd = 43;
016819  timbreCmd = 44;
016820  getAmpCmd = 45;
016821
016822  waveTableCmd = 60;
016823  phaseCmd = 61;
016824
016825  soundCmd = 80;
016826  bufferCmd = 81;
016827  rateCmd = 82;
016828  continueCmd = 83;
016829  doubleBufferCmd = 84;
016830  getRateCmd = 85;
016831
016832  sizeCmd = 90;
016833  convertCmd = 91;
016834
016835  stdQLength = 128;
016836  dataOffsetFlag = $8000;
016837
016838  waveInitChannelMask = $07;
016839  waveInitChannel0 = $04;
```

```
016840  waveInitChannel1 = $05;
016841  waveInitChannel2 = $06;
016842  waveInitChannel3 = $07;
016843
016844  { channel initialization parameters }
016845  initPanMask = $0003;                    { mask for right/left pan values }
016846  initSRateMask = $0030;                  { mask for sample rate values }
016847  initStereoMask = $00C0;                 { mask for mono/stereo values }
016848  initCompMask = $FF00;                   { mask for compression IDs }
016849
016850  initChanLeft = $0002;                   { left stereo channel }
016851  initChanRight = $0003;                  { right stereo channel }
016852  initNoInterp = $0004;                   { no linear interpolation }
016853  initNoDrop = $0008;                     { no drop-sample conversion }
016854  initMono = $0080;                       { monophonic channel }
016855  initStereo = $00C0;                     { stereo channel }
016856  initMACE3 = $0300;                      { MACE 3:1 }
016857  initMACE6 = $0400;                      { MACE 6:1 }
016858
016859  initChan0 = $0004;                      { channel 0 - wave table only }
016860  initChan1 = $0005;                      { channel 1 - wave table only }
016861  initChan2 = $0006;                      { channel 2 - wave table only }
016862  initChan3 = $0007;                      { channel 3 - wave table only }
016863
016864  stdSH = $00;                            { Standard sound header encode value }
016865  extSH = $FF;                            { Extended sound header encode value }
016866  cmpSH = $FE;                            { Compressed sound header encode value }
016867
016868  notCompressed = 0;                      { compression ID's }
016869  twoToOne = 1;
016870  eightToThree = 2;
016871  threeToOne = 3;
016872  sixToOne = 4;
016873
016874  outsideCmpSH = 0;                       { MACE constants }
016875  insideCmpSH = 1;
016876  aceSuccess = 0;
016877  aceMemFull = 1;
016878  aceNilBlock = 2;
016879  aceBadComp = 3;
016880  aceBadEncode = 4;
016881  aceBadDest = 5;
016882  aceBadCmd = 6;
016883  sixToOnePacketSize = 8;
016884  threeToOnePacketSize = 16;
016885  stateBlockSize = 64;
016886  leftOverBlockSize = 32;
016887
016888  firstSoundFormat = $0001;               { general sound format }
016889  secondSoundFormat = $0002;              { special sampled sound format (HyperCard) }
016890
016891  dbBufferReady = $00000001;              { double buffer is filled }
016892  dbLastBuffer = $00000004;               { last double buffer to play }
016893
016894  sysBeepDisable = $0000;                 { SysBeep() enable flags }
016895  sysBeepEnable = $0001;
```

```
016896
016897  unitTypeNoSelection = $FFFF;               { unitTypes for AudioSelection.unitType }
016898  unitTypeSeconds = $0000;
016899
016900  TYPE
016901  {   Structures for Sound Driver   }
016902
016903
016904  FreeWave = PACKED ARRAY [0..30000] OF Byte;
016905
016906  FFSynthPtr = ^FFSynthRec;
016907  FFSynthRec = RECORD
016908   mode: INTEGER;
016909   count: Fixed;
016910   waveBytes: FreeWave;
016911   END;
016912
016913  Tone = RECORD
016914   count: INTEGER;
016915   amplitude: INTEGER;
016916   duration: INTEGER;
016917   END;
016918
016919
016920  Tones = ARRAY [0..5000] OF Tone;
016921
016922  SWSynthPtr = ^SWSynthRec;
016923  SWSynthRec = RECORD
016924   mode: INTEGER;
016925   triplets: Tones;
016926   END;
016927
016928
016929  Wave = PACKED ARRAY [0..255] OF Byte;
016930  WavePtr = ^Wave;
016931
016932  FTSndRecPtr = ^FTSoundRec;
016933  FTSoundRec = RECORD
016934   duration: INTEGER;
016935   sound1Rate: Fixed;
016936   sound1Phase: LONGINT;
016937   sound2Rate: Fixed;
016938   sound2Phase: LONGINT;
016939   sound3Rate: Fixed;
016940   sound3Phase: LONGINT;
016941   sound4Rate: Fixed;
016942   sound4Phase: LONGINT;
016943   sound1Wave: WavePtr;
016944   sound2Wave: WavePtr;
016945   sound3Wave: WavePtr;
016946   sound4Wave: WavePtr;
016947   END;
016948
016949  FTSynthPtr = ^FTSynthRec;
016950  FTSynthRec = RECORD
016951   mode: INTEGER;
```

```
016952   sndRec: FTSndRecPtr;
016953   END;
016954
016955  {   Structures for Sound Manager   }
016956
016957  SndCommand = PACKED RECORD
016958   cmd: INTEGER;
016959   param1: INTEGER;
016960   param2: LONGINT;
016961   END;
016962
016963
016964  Time = LONGINT;                              { in half milliseconds }
016965
016966
016967
016968  SndChannelPtr = ^SndChannel;
016969  SndChannel = PACKED RECORD
016970   nextChan: SndChannelPtr;
016971   firstMod: Ptr;                              { reserved for the Sound Manager }
016972   callBack: ProcPtr;
016973   userInfo: LONGINT;
016974   wait: Time;                                 { The following is for internal Sound Manager use only.}
016975   cmdInProgress: SndCommand;
016976   flags: INTEGER;
016977   qLength: INTEGER;
016978   qHead: INTEGER;                             { next spot to read or -1 if empty }
016979   qTail: INTEGER;                             { next spot to write = qHead if full }
016980   queue: ARRAY [0..stdQLength - 1] OF SndCommand;
016981   END;
016982
016983  { MACE structures }
016984  StateBlockPtr = ^StateBlock;
016985  StateBlock = RECORD
016986   stateVar: ARRAY [0..stateBlockSize - 1] OF INTEGER;
016987   END;
016988
016989  LeftOverBlockPtr = ^LeftOverBlock;
016990  LeftOverBlock = RECORD
016991   count: LONGINT;
016992   sampleArea: PACKED ARRAY [0..leftOverBlockSize - 1] OF Byte;
016993   END;
016994
016995  ModRef = RECORD
016996   modNumber: INTEGER;
016997   modInit: LONGINT;
016998   END;
016999
017000  SndListPtr = ^SndListResource;
017001  SndListResource = RECORD
017002   format: INTEGER;
017003   numModifiers: INTEGER;
017004   modifierPart: ARRAY [0..0] OF ModRef;       {This is a variable length array}
017005   numCommands: INTEGER;
017006   commandPart: ARRAY [0..0] OF SndCommand;    {This is a variable length array}
017007   dataPart: PACKED ARRAY [0..0] OF Byte;      {This is a variable length array}
```

```
017008   END;
017009
017010  SoundHeaderPtr = ^SoundHeader;
017011  SoundHeader = PACKED RECORD
017012   samplePtr: Ptr;                              { if NIL then samples are in sampleArea }
017013   length: LONGINT;                             { length of sound in bytes }
017014   sampleRate: Fixed;                           { sample rate for this sound }
017015   loopStart: LONGINT;                          { start of looping portion }
017016   loopEnd: LONGINT;                            { end of looping portion }
017017   encode: Byte;                                { header encoding }
017018   baseFrequency: Byte;                         { baseFrequency value }
017019   sampleArea: PACKED ARRAY [0..0] OF Byte;
017020   END;
017021
017022  CmpSoundHeaderPtr = ^CmpSoundHeader;
017023  CmpSoundHeader = PACKED RECORD
017024   samplePtr: Ptr;                              { if nil then samples are in sample area }
017025   numChannels: LONGINT;                        { number of channels i.e. mono = 1 }
017026   sampleRate: Fixed;                           { sample rate in Apples Fixed point representation }
017027   loopStart: LONGINT;                          { loopStart of sound before compression }
017028   loopEnd: LONGINT;                            { loopEnd of sound before compression }
017029   encode: Byte;                                { data structure used , stdSH, extSH, or cmpSH }
017030   baseFrequency: Byte;                         { same meaning as regular SoundHeader }
017031   numFrames: LONGINT;                          { length in frames ( packetFrames or sampleFrames ) }
017032   AIFFSampleRate: Extended80;                  { IEEE sample rate }
017033   markerChunk: Ptr;                            { sync track }
017034   futureUse1: Ptr;                             { reserved by Apple }
017035   futureUse2: Ptr;                             { reserved by Apple }
017036   stateVars: StateBlockPtr;                    { pointer to State Block }
017037   leftOverSamples: LeftOverBlockPtr;           { used to save truncated samples between compression calls }
017038   compressionID: INTEGER;                      { 0 means no compression, non zero means compressionID }
017039   packetSize: INTEGER;                         { number of bits in compressed sample packet }
017040   snthID: INTEGER;                             { resource ID of Sound Manager snth that contains NRT C/E }
017041   sampleSize: INTEGER;                         { number of bits in non-compressed sample }
017042   sampleArea: PACKED ARRAY [0..0] OF Byte;     { space for when samples follow directly }
017043   END;
017044
017045  ExtSoundHeaderPtr = ^ExtSoundHeader;
017046  ExtSoundHeader = PACKED RECORD
017047   samplePtr: Ptr;                              { if nil then samples are in sample area }
017048   numChannels: LONGINT;                        { number of channels,  ie mono = 1 }
017049   sampleRate: Fixed;                           { sample rate in Apples Fixed point representation }
017050   loopStart: LONGINT;                          { same meaning as regular SoundHeader }
017051   loopEnd: LONGINT;                            { same meaning as regular SoundHeader }
017052   encode: Byte;                                { data structure used , stdSH, extSH, or cmpSH }
017053   baseFrequency: Byte;                         { same meaning as regular SoundHeader }
017054   numFrames: LONGINT;                          { length in total number of frames }
017055   AIFFSampleRate: Extended80;                  { IEEE sample rate }
017056   markerChunk: Ptr;                            { sync track }
017057   instrumentChunks: Ptr;                       { AIFF instrument chunks }
017058   AESRecording: Ptr;
017059   sampleSize: INTEGER;                         { number of bits in sample }
017060   futureUse1: INTEGER;                         { reserved by Apple }
017061   futureUse2: LONGINT;                         { reserved by Apple }
017062   futureUse3: LONGINT;                         { reserved by Apple }
017063   futureUse4: LONGINT;                         { reserved by Apple }
```

```
017064   sampleArea: PACKED ARRAY [0..0] OF Byte;   { space for when samples follow directly }
017065   END;
017066
017067  ConversionBlockPtr = ^ConversionBlock;
017068  ConversionBlock = RECORD
017069   destination: INTEGER;
017070   unused: INTEGER;
017071   inputPtr: CmpSoundHeaderPtr;
017072   outputPtr: CmpSoundHeaderPtr;
017073   END;
017074
017075  SMStatusPtr = ^SMStatus;
017076  SMStatus = PACKED RECORD
017077   smMaxCPULoad: INTEGER;
017078   smNumChannels: INTEGER;
017079   smCurCPULoad: INTEGER;
017080   END;
017081
017082  SCStatusPtr = ^SCStatus;
017083  SCStatus = RECORD
017084   scStartTime: Fixed;
017085   scEndTime: Fixed;
017086   scCurrentTime: Fixed;
017087   scChannelBusy: BOOLEAN;
017088   scChannelDisposed: BOOLEAN;
017089   scChannelPaused: BOOLEAN;
017090   scUnused: BOOLEAN;
017091   scChannelAttributes: LONGINT;
017092   scCPULoad: LONGINT;
017093   END;
017094
017095  AudioSelectionPtr = ^AudioSelection;
017096  AudioSelection = PACKED RECORD
017097   unitType: LONGINT;
017098   selStart: Fixed;
017099   selEnd: Fixed;
017100   END;
017101
017102  SndDoubleBufferPtr = ^SndDoubleBuffer;
017103  SndDoubleBuffer = PACKED RECORD
017104   dbNumFrames: LONGINT;
017105   dbFlags: LONGINT;
017106   dbUserInfo: ARRAY [0..1] OF LONGINT;
017107   dbSoundData: PACKED ARRAY [0..0] OF Byte;
017108   END;
017109
017110  SndDoubleBufferHeaderPtr = ^SndDoubleBufferHeader;
017111  SndDoubleBufferHeader = PACKED RECORD
017112   dbhNumChannels: INTEGER;
017113   dbhSampleSize: INTEGER;
017114   dbhCompressionID: INTEGER;
017115   dbhPacketSize: INTEGER;
017116   dbhSampleRate: Fixed;
017117   dbhBufferPtr: ARRAY [0..1] OF SndDoubleBufferPtr;
017118   dbhDoubleBack: ProcPtr;
017119   END;
```

```
017120
017121
017122  FUNCTION SndDoCommand(chan: SndChannelPtr;cmd: SndCommand;noWait: BOOLEAN): OSErr;
017123   INLINE $A803;
017124  FUNCTION SndDoImmediate(chan: SndChannelPtr;cmd: SndCommand): OSErr;
017125   INLINE $A804;
017126  FUNCTION SndNewChannel(VAR chan: SndChannelPtr;synth: INTEGER;init: LONGINT;
017127   userRoutine: ProcPtr): OSErr;
017128   INLINE $A807;
017129  FUNCTION SndDisposeChannel(chan: SndChannelPtr;quietNow: BOOLEAN): OSErr;
017130   INLINE $A801;
017131  FUNCTION SndPlay(chan: SndChannelPtr;sndHdl: Handle;async: BOOLEAN): OSErr;
017132   INLINE $A805;
017133  FUNCTION SndAddModifier(chan: SndChannelPtr;modifier: ProcPtr;id: INTEGER;
017134   init: LONGINT): OSErr;
017135   INLINE $A802;
017136  FUNCTION SndControl(id: INTEGER;VAR cmd: SndCommand): OSErr;
017137   INLINE $A806;
017138
017139  PROCEDURE SetSoundVol(level: INTEGER);
017140  PROCEDURE GetSoundVol(VAR level: INTEGER);
017141  PROCEDURE StartSound(synthRec: Ptr;numBytes: LONGINT;completionRtn: ProcPtr);
017142  PROCEDURE StopSound;
017143  FUNCTION SoundDone: BOOLEAN;
017144
017145  FUNCTION SndSoundManagerVersion: NumVersion;
017146   INLINE $203C,$000C,$0008,$A800;
017147  FUNCTION SndStartFilePlay(chan: SndChannelPtr;fRefNum: INTEGER;resNum: INTEGER;
017148   bufferSize: LONGINT;theBuffer: Ptr;theSelection: AudioSelectionPtr;theCompletion: ProcPtr;
017149   async: BOOLEAN): OSErr;
017150   INLINE $203C,$0D00,$0008,$A800;
017151  FUNCTION SndPauseFilePlay(chan: SndChannelPtr): OSErr;
017152   INLINE $203C,$0204,$0008,$A800;
017153  FUNCTION SndStopFilePlay(chan: SndChannelPtr;async: BOOLEAN): OSErr;
017154   INLINE $203C,$0308,$0008,$A800;
017155  FUNCTION SndChannelStatus(chan: SndChannelPtr;theLength: INTEGER;theStatus: SCStatusPtr): OSErr;
017156   INLINE $203C,$0010,$0008,$A800;
017157  FUNCTION SndManagerStatus(theLength: INTEGER;theStatus: SMStatusPtr): OSErr;
017158   INLINE $203C,$0014,$0008,$A800;
017159  PROCEDURE SndGetSysBeepState(VAR sysBeepState: INTEGER);
017160   INLINE $203C,$0018,$0008,$A800;
017161  FUNCTION SndSetSysBeepState(sysBeepState: INTEGER): OSErr;
017162   INLINE $203C,$001C,$0008,$A800;
017163  FUNCTION SndPlayDoubleBuffer(chan: SndChannelPtr;theParams: SndDoubleBufferHeaderPtr): OSErr;
017164   INLINE $203C,$0020,$0008,$A800;
017165
017166  FUNCTION MACEVersion: NumVersion;
017167   INLINE $203C,$0000,$0010,$A800;
017168  PROCEDURE Comp3to1(inBuffer: Ptr;outBuffer: Ptr;cnt: LONGINT;inState: Ptr;
017169   outState: Ptr;numChannels: LONGINT;whichChannel: LONGINT);
017170   INLINE $203C,$0004,$0010,$A800;
017171  PROCEDURE Exp1to3(inBuffer: Ptr;outBuffer: Ptr;cnt: LONGINT;inState: Ptr;
017172   outState: Ptr;numChannels: LONGINT;whichChannel: LONGINT);
017173   INLINE $203C,$0008,$0010,$A800;
017174  PROCEDURE Comp6to1(inBuffer: Ptr;outBuffer: Ptr;cnt: LONGINT;inState: Ptr;
017175   outState: Ptr;numChannels: LONGINT;whichChannel: LONGINT);
```

```
017176   INLINE $203C,$000C,$0010,$A800;
017177  PROCEDURE Exp1to6(inBuffer: Ptr;outBuffer: Ptr;cnt: LONGINT;inState: Ptr;
017178   outState: Ptr;numChannels: LONGINT;whichChannel: LONGINT);
017179   INLINE $203C,$0010,$0010,$A800;
017180
017181
017182  {$ENDC} { UsingSound }
017183
017184  {$IFC NOT UsingIncludes}
017185   END.
017186  {$ENDC}
017187
017188
017189  ### END OF FILE Sound.p
017190
```

```
017191
017192  ############################################################################
017193  ### FILE: SoundInput.p
017194  ############################################################################
017195
017196
017197  {
017198  Created: Tuesday, September 10, 1991 at 2:15 PM
017199   SoundInput.p
017200   Pascal Interface to the Macintosh Libraries
017201
017202    Copyright Apple Computer, Inc. 1990-1991
017203    All rights reserved
017204  }
017205
017206
017207  {$IFC UNDEFINED UsingIncludes}
017208  {$SETC UsingIncludes := 0}
017209  {$ENDC}
017210
017211  {$IFC NOT UsingIncludes}
017212   UNIT SoundInput;
017213   INTERFACE
017214  {$ENDC}
017215
017216  {$IFC UNDEFINED UsingSoundInput}
017217  {$SETC UsingSoundInput := 1}
017218
017219  {$I+}
017220  {$SETC SoundInputIncludes := UsingIncludes}
017221  {$SETC UsingIncludes := 1}
017222  {$IFC UNDEFINED UsingTypes}
017223  {$I $$Shell(PInterfaces)Types.p}
017224  {$ENDC}
017225  {$IFC UNDEFINED UsingDialogs}
017226  {$I $$Shell(PInterfaces)Dialogs.p}
017227  {$ENDC}
017228  {$IFC UNDEFINED UsingFiles}
017229  {$I $$Shell(PInterfaces)Files.p}
017230  {$ENDC}
017231  {$SETC UsingIncludes := SoundInputIncludes}
017232
017233  CONST
017234  siDeviceIsConnected = 1;            { input device is connected and ready for input }
017235  siDeviceNotConnected = 0;           { input device is not connected }
017236  siDontKnowIfConnected = -1;         { can't tell if input device is connected }
017237  siReadPermission = 0;               { permission passed to SPBOpenDevice }
017238  siWritePermission = 1;              { permission passed to SPBOpenDevice }
017239
017240  { Info Selectors for Sound Input Drivers }
017241  siDeviceConnected = 'dcon';         { input device connection status }
017242  siAGCOnOff = 'agc ';                { automatic gain control state }
017243  siPlayThruOnOff = 'plth';           { playthrough state }
017244  siTwosComplementOnOff = 'twos';     { two's complement state }
017245  siLevelMeterOnOff = 'lmet';         { level meter state }
017246  siRecordingQuality = 'qual';        { recording quality }
```

```
017247  siVoxRecordInfo = 'voxr';           { VOX record parameters }
017248  siVoxStopInfo = 'voxs';             { VOX stop parameters }
017249  siNumberChannels = 'chan';          { current number of channels }
017250  siSampleSize = 'ssiz';              { current sample size }
017251  siSampleRate = 'srat';              { current sample rate }
017252  siCompressionType = 'comp';         { current compression type }
017253  siCompressionFactor = 'cmfa';       { current compression factor }
017254  siCompressionHeader = 'cmhd';       { return compression header }
017255  siDeviceName = 'name';              { input device name }
017256  siDeviceIcon = 'icon';              { input device icon }
017257  siDeviceBufferInfo = 'dbin';        { size of interrupt buffer }
017258  siSampleSizeAvailable = 'ssav';     { sample sizes available }
017259  siSampleRateAvailable = 'srav';     { sample rates available }
017260  siCompressionAvailable = 'cmav';    { compression types available }
017261  siChannelAvailable = 'chav';        { number of channels available }
017262  siAsync = 'asyn';                   { asynchronous capability }
017263  siOptionsDialog = 'optd';           { display options dialog }
017264  siContinuous = 'cont';              { continous recording }
017265  siActiveChannels = 'chac';          { active channels }
017266  siActiveLevels = 'lmac';            { active meter levels }
017267  siInputSource = 'sour';             { input source selector }
017268  siInitializeDriver = 'init';        { reserved for internal use only }
017269  siCloseDriver = 'clos';             { reserved for internal use only }
017270  siPauseRecording = 'paus';          { reserved for internal use only }
017271  siUserInterruptProc = 'user';       { reserved for internal use only }
017272
017273  { Qualities }
017274  siBestQuality = 'best';
017275  siBetterQuality = 'betr';
017276  siGoodQuality = 'good';
017277
017278  TYPE
017279  { Sound Input Parameter Block }
017280  SPBPtr = ^SPB;
017281  SPB = RECORD
017282   inRefNum: LONGINT;                 { reference number of sound input device }
017283   count: LONGINT;                    { number of bytes to record }
017284   milliseconds: LONGINT;             { number of milliseconds to record }
017285   bufferLength: LONGINT;             { length of buffer in bytes }
017286   bufferPtr: Ptr;                    { buffer to store sound data in }
017287   completionRoutine: ProcPtr;        { completion routine }
017288   interruptRoutine: ProcPtr;         { interrupt routine }
017289   userLong: LONGINT;                 { user-defined field }
017290   error: OSErr;                      { error }
017291   unused1: LONGINT;                  { reserved - must be zero }
017292   END;
017293
017294
017295  FUNCTION SPBVersion: NumVersion;
017296   INLINE $203C,$0000,$0014,$A800;
017297  FUNCTION SndRecord(filterProc: ModalFilterProcPtr;corner: Point;quality: OSType;
017298   VAR sndHandle: Handle): OSErr;
017299   INLINE $203C,$0804,$0014,$A800;
017300  FUNCTION SndRecordToFile(filterProc: ModalFilterProcPtr;corner: Point;quality: OSType;
017301   fRefNum: INTEGER): OSErr;
017302   INLINE $203C,$0708,$0014,$A800;
```

```
017303  FUNCTION SPBSignInDevice(deviceRefNum: INTEGER;deviceName: Str255): OSErr;
017304   INLINE $203C,$030C,$0014,$A800;
017305  FUNCTION SPBSignOutDevice(deviceRefNum: INTEGER): OSErr;
017306   INLINE $203C,$0110,$0014,$A800;
017307  FUNCTION SPBGetIndexedDevice(count: INTEGER;VAR deviceName: Str255;VAR deviceIconHandle: Handle): OSErr;
017308   INLINE $203C,$0514,$0014,$A800;
017309  FUNCTION SPBOpenDevice(deviceName: Str255;permission: INTEGER;VAR inRefNum: LONGINT): OSErr;
017310   INLINE $203C,$0518,$0014,$A800;
017311  FUNCTION SPBCloseDevice(inRefNum: LONGINT): OSErr;
017312   INLINE $203C,$021C,$0014,$A800;
017313  FUNCTION SPBRecord(inParamPtr: SPBPtr;asynchFlag: BOOLEAN): OSErr;
017314   INLINE $203C,$0320,$0014,$A800;
017315  FUNCTION SPBRecordToFile(fRefNum: INTEGER;inParamPtr: SPBPtr;asynchFlag: BOOLEAN): OSErr;
017316   INLINE $203C,$0424,$0014,$A800;
017317  FUNCTION SPBPauseRecording(inRefNum: LONGINT): OSErr;
017318   INLINE $203C,$0228,$0014,$A800;
017319  FUNCTION SPBResumeRecording(inRefNum: LONGINT): OSErr;
017320   INLINE $203C,$022C,$0014,$A800;
017321  FUNCTION SPBStopRecording(inRefNum: LONGINT): OSErr;
017322   INLINE $203C,$0230,$0014,$A800;
017323  FUNCTION SPBGetRecordingStatus(inRefNum: LONGINT;VAR recordingStatus: INTEGER;
017324   VAR meterLevel: INTEGER;VAR totalSamplesToRecord: LONGINT;VAR numberOfSamplesRecorded: LONGINT;
017325   VAR totalMsecsToRecord: LONGINT;VAR numberOfMsecsRecorded: LONGINT): OSErr;
017326   INLINE $203C,$0E34,$0014,$A800;
017327  FUNCTION SPBGetDeviceInfo(inRefNum: LONGINT;infoType: OSType;infoData: Ptr): OSErr;
017328   INLINE $203C,$0638,$0014,$A800;
017329  FUNCTION SPBSetDeviceInfo(inRefNum: LONGINT;infoType: OSType;infoData: Ptr): OSErr;
017330   INLINE $203C,$063C,$0014,$A800;
017331  FUNCTION SPBMillisecondsToBytes(inRefNum: LONGINT;VAR milliseconds: LONGINT): OSErr;
017332   INLINE $203C,$0440,$0014,$A800;
017333  FUNCTION SPBBytesToMilliseconds(inRefNum: LONGINT;VAR byteCount: LONGINT): OSErr;
017334   INLINE $203C,$0444,$0014,$A800;
017335  FUNCTION SetupSndHeader(sndHandle: Handle;numChannels: INTEGER;sampleRate: Fixed;
017336   sampleSize: INTEGER;compressionType: OSType;baseNote: INTEGER;numBytes: LONGINT;
017337   VAR headerLen: INTEGER): OSErr;
017338   INLINE $203C,$0D48,$0014,$A800;
017339  FUNCTION SetupAIFFHeader(fRefNum: INTEGER;numChannels: INTEGER;sampleRate: Fixed;
017340   sampleSize: INTEGER;compressionType: OSType;numBytes: LONGINT;numFrames: LONGINT): OSErr;
017341   INLINE $203C,$0B4C,$0014,$A800;
017342
017343
017344  {$ENDC} { UsingSoundInput }
017345
017346  {$IFC NOT UsingIncludes}
017347   END.
017348  {$ENDC}
017349
017350
017351  ### END OF FILE SoundInput.p
017352
```

```
017353
017354  ##########################################################################
017355  ### FILE: StandardFile.p
017356  ##########################################################################
017357
017358  {
017359  Created: Sunday, January 6, 1991 at 11:19 PM
017360      StandardFile.p
017361      Pascal Interface to the Macintosh Libraries
017362
017363          Copyright Apple Computer, Inc.    1990
017364          All rights reserved
017365
017366  }
017367
017368
017369  {$IFC UNDEFINED UsingIncludes}
017370  {$SETC UsingIncludes := 0}
017371  {$ENDC}
017372
017373  {$IFC NOT UsingIncludes}
017374      UNIT StandardFile;
017375      INTERFACE
017376  {$ENDC}
017377
017378  {$IFC UNDEFINED UsingStandardFile}
017379  {$SETC UsingStandardFile := 1}
017380
017381  {$I+}
017382  {$SETC StandardFileIncludes := UsingIncludes}
017383  {$SETC UsingIncludes := 1}
017384  {$IFC UNDEFINED UsingTypes}
017385  {$I $$Shell(PInterfaces)Types.p}
017386  {$ENDC}
017387  {$IFC UNDEFINED UsingDialogs}
017388  {$I $$Shell(PInterfaces)Dialogs.p}
017389  {$ENDC}
017390  {$IFC UNDEFINED UsingFiles}
017391  {$I $$Shell(PInterfaces)Files.p}
017392  {$ENDC}
017393  {$SETC UsingIncludes := StandardFileIncludes}
017394
017395  CONST
017396
017397  { resource IDs and item offsets of pre-7.0 dialogs }
017398  putDlgID = -3999;
017399  putSave = 1;
017400  putCancel = 2;
017401  putEject = 5;
017402  putDrive = 6;
017403  putName = 7;
017404
017405  getDlgID = -4000;
017406  getOpen = 1;
017407  getCancel = 3;
017408  getEject = 5;
```

```
017409  getDrive = 6;
017410  getNmList = 7;
017411  getScroll = 8;
017412
017413  { resource IDs and item offsets of 7.0 dialogs }
017414  sfPutDialogID = -6043;
017415  sfGetDialogID = -6042;
017416  sfItemOpenButton = 1;
017417  sfItemCancelButton = 2;
017418  sfItemBalloonHelp = 3;
017419  sfItemVolumeUser = 4;
017420  sfItemEjectButton = 5;
017421  sfItemDesktopButton = 6;
017422  sfItemFileListUser = 7;
017423  sfItemPopUpMenuUser = 8;
017424  sfItemDividerLinePict = 9;
017425  sfItemFileNameTextEdit = 10;
017426  sfItemPromptStaticText = 11;
017427  sfItemNewFolderUser = 12;
017428
017429
017430  { pseudo-item hits for use in DlgHook }
017431  sfHookFirstCall = -1;
017432  sfHookCharOffset = $1000;
017433  sfHookNullEvent = 100;
017434  sfHookRebuildList = 101;
017435  sfHookFolderPopUp = 102;
017436  sfHookOpenFolder = 103;
017437
017438
017439  { the following are only in system 7.0+ }
017440  sfHookOpenAlias = 104;
017441  sfHookGoToDesktop = 105;
017442  sfHookGoToAliasTarget = 106;
017443  sfHookGoToParent = 107;
017444  sfHookGoToNextDrive = 108;
017445  sfHookGoToPrevDrive = 109;
017446  sfHookChangeSelection = 110;
017447  sfHookSetActiveOffset = 200;
017448  sfHookLastCall = -2;
017449
017450  { the refcon field of the dialog record during a
017451   modalfilter or dialoghook contains one of the following }
017452  sfMainDialogRefCon = 'stdf';
017453  sfNewFolderDialogRefCon = 'nfdr';
017454  sfReplaceDialogRefCon = 'rplc';
017455  sfStatWarnDialogRefCon = 'stat';
017456  sfLockWarnDialogRefCon = 'lock';
017457  sfErrorDialogRefCon = 'err ';
017458
017459  TYPE
017460  SFReply = RECORD
017461      good: BOOLEAN;
017462      copy: BOOLEAN;
017463      fType: OSType;
017464      vRefNum: INTEGER;
```

```
017465      version: INTEGER;
017466      fName: Str63;
017467      END;
017468
017469  StandardFileReply = RECORD
017470      sfGood: BOOLEAN;
017471      sfReplacing: BOOLEAN;
017472      sfType: OSType;
017473      sfFile: FSSpec;
017474      sfScript: ScriptCode;
017475      sfFlags: INTEGER;
017476      sfIsFolder: BOOLEAN;
017477      sfIsVolume: BOOLEAN;
017478      sfReserved1: LONGINT;
017479      sfReserved2: INTEGER;
017480      END;
017481
017482
017483  DlgHookProcPtr = ProcPtr;           { FUNCTION Hook(item: INTEGER; theDialog: DialogPtr): INTEGER; }
017484  FileFilterProcPtr = ProcPtr;        { FUNCTION FileFilter(PB: CInfoPBPtr): BOOLEAN; }
017485
017486  { the following also include an extra parameter of "your data pointer" }
017487  DlgHookYDProcPtr = ProcPtr;         { FUNCTION  Hook(item: INTEGER; theDialog: DialogPtr; yourDataPtr: Ptr): INTEGER; }
017488  ModalFilterYDProcPtr = ProcPtr;     { FUNCTION  Filter(theDialog: DialogPtr; VAR theEvent: EventRecord; VAR itemHit: INTEGER; yourDataPtr: Ptr): B(
017489  FileFilterYDProcPtr = ProcPtr;      { FUNCTION  FileFilter(PB: CInfoPBPtr; yourDataPtr: Ptr): BOOLEAN; }
017490  ActivateYDProcPtr = ProcPtr;        { PROCEDURE Activate(theDialog; DialogPtr; itemNo: INTEGER; activating: BOOLEAN; yourDataPtr: Ptr); }
017491
017492  SFTypeList = ARRAY [0..3] OF OSType;
017493
017494  PROCEDURE SFPutFile(where: Point;
017495                      prompt: Str255;
017496                      origName: Str255;
017497                      dlgHook: DlgHookProcPtr;
017498                      VAR reply: SFReply);
017499      INLINE $3F3C,$0001,$A9EA;
017500
017501  PROCEDURE SFGetFile(where: Point;
017502                      prompt: Str255;
017503                      fileFilter: FileFilterProcPtr;
017504                      numTypes: INTEGER;
017505                      typeList: SFTypeList;
017506                      dlgHook: DlgHookProcPtr;
017507                      VAR reply: SFReply);
017508      INLINE $3F3C,$0002,$A9EA;
017509
017510  PROCEDURE SFPPutFile(where: Point;
017511                       prompt: Str255;
017512                       origName: Str255;
017513                       dlgHook: DlgHookProcPtr;
017514                       VAR reply: SFReply;
017515                       dlgID: INTEGER;
017516                       filterProc: ModalFilterProcPtr);
017517      INLINE $3F3C,$0003,$A9EA;
017518
017519  PROCEDURE SFPGetFile(where: Point;
017520                       prompt: Str255;
```

```
017521                      fileFilter: FileFilterProcPtr;
017522                      numTypes: INTEGER;
017523                      typeList: SFTypeList;
017524                      dlgHook: DlgHookProcPtr;
017525                      VAR reply: SFReply;
017526                      dlgID: INTEGER;
017527                      filterProc: ModalFilterProcPtr);
017528      INLINE $3F3C,$0004,$A9EA;
017529
017530  PROCEDURE StandardPutFile(prompt: Str255;
017531                            defaultName: Str255;
017532                            VAR reply: StandardFileReply);
017533      INLINE $3F3C,$0005,$A9EA;
017534
017535  PROCEDURE StandardGetFile(fileFilter: FileFilterProcPtr;
017536                            numTypes: INTEGER;
017537                            typeList: SFTypeList;
017538                            VAR reply: StandardFileReply);
017539      INLINE $3F3C,$0006,$A9EA;
017540
017541  PROCEDURE CustomPutFile(prompt: Str255;
017542                          defaultName: Str255;
017543                          VAR reply: StandardFileReply;
017544                          dlgID: INTEGER;
017545                          where: Point;
017546                          dlgHook: DlgHookYDProcPtr;
017547                          filterProc: ModalFilterYDProcPtr;
017548                          activeList: Ptr;
017549                          activateProc: ActivateYDProcPtr;
017550                          yourDataPtr: UNIV Ptr);
017551      INLINE $3F3C,$0007,$A9EA;
017552
017553  PROCEDURE CustomGetFile(fileFilter: FileFilterYDProcPtr;
017554                          numTypes: INTEGER;
017555                          typeList: SFTypeList;
017556                          VAR reply: StandardFileReply;
017557                          dlgID: INTEGER;
017558                          where: Point;
017559                          dlgHook: DlgHookYDProcPtr;
017560                          filterProc: ModalFilterYDProcPtr;
017561                          activeList: Ptr;
017562                          activateProc: ActivateYDProcPtr;
017563                          yourDataPtr: UNIV Ptr);
017564      INLINE $3F3C,$0008,$A9EA;
017565
017566
017567  {
017568      New StandardFile routine comments:
017569
017570      activeList is pointer to array of integer (16-bits).
017571      first integer is length of list.
017572      following integers are possible activatable DITL items, in
017573      the order that the tab key will cycle through.  The first
017574      in the list is the item made active when dialog is first shown.
017575
017576      activateProc is a pointer to a procedure like:
```

```
017577
017578          PROCEDURE MyActivateProc(theDialog:     DialogPtr;
017579                                   itemNo:        INTEGER;
017580                                   activating:    BOOLEAN;
017581                                   yourDataPtr:    Ptr);
017582
017583      The activateProc is called with activating=FALSE on the itemNo
017584      about to deactivate then with activating=TRUE on the itemNo
017585      about to become the active item. (like activate event)
017586
017587      yourDataPtr is a nice little extra that makes life easier without
017588      globals.  CustomGetFile & CustomPPutFile when calling any of their
017589      call back procedures, pushes the extra parameter of yourDataPtr on
017590      the stack.
017591
017592      In addition the filterProc in CustomGetFile & CustomPPutFile is called
017593      before before SF does any mapping, instead of after.
017594   }
017595
017596
017597
017598  {$ENDC}    { UsingStandardFile }
017599
017600  {$IFC NOT UsingIncludes}
017601      END.
017602  {$ENDC}
017603
017604
017605  ### END OF FILE StandardFile.p
017606
```

```
017607
017608  ############################################################################
017609  ### FILE: Start.p
017610  ############################################################################
017611
017612  {
017613  Created: Sunday, January 6, 1991 at 11:20 PM
017614      Start.p
017615      Pascal Interface to the Macintosh Libraries
017616
017617          Copyright Apple Computer, Inc.    1987-1990
017618          All rights reserved
017619  }
017620
017621
017622  {$IFC UNDEFINED UsingIncludes}
017623  {$SETC UsingIncludes := 0}
017624  {$ENDC}
017625
017626  {$IFC NOT UsingIncludes}
017627      UNIT Start;
017628      INTERFACE
017629  {$ENDC}
017630
017631  {$IFC UNDEFINED UsingStart}
017632  {$SETC UsingStart := 1}
017633
017634  {$I+}
017635  {$SETC StartIncludes := UsingIncludes}
017636  {$SETC UsingIncludes := 1}
017637  {$IFC UNDEFINED UsingTypes}
017638  {$I $$Shell(PInterfaces)Types.p}
017639  {$ENDC}
017640  {$SETC UsingIncludes := StartIncludes}
017641
017642  TYPE
017643  DefStartType = (slotDev,scsiDev);
017644
017645
017646  DefStartPtr = ^DefStartRec;
017647  DefStartRec = RECORD
017648      CASE DefStartType OF
017649        slotDev:
017650          (sdExtDevID: SignedByte;
017651          sdPartition: SignedByte;
017652          sdSlotNum: SignedByte;
017653          sdSRsrcID: SignedByte);
017654        scsiDev:
017655          (sdReserved1: SignedByte;
017656          sdReserved2: SignedByte;
017657          sdRefNum: INTEGER);
017658      END;
017659
017660  DefVideoPtr = ^DefVideoRec;
017661  DefVideoRec = RECORD
017662      sdSlot: SignedByte;
```

```
017663      sdsResource: SignedByte;
017664      END;
017665
017666  DefOSPtr = ^DefOSRec;
017667  DefOSRec = RECORD
017668      sdReserved: SignedByte;
017669      sdOSType: SignedByte;
017670      END;
017671
017672
017673  PROCEDURE GetDefaultStartup(paramBlock: DefStartPtr);
017674      INLINE $205F,$A07D;
017675  PROCEDURE SetDefaultStartup(paramBlock: DefStartPtr);
017676      INLINE $205F,$A07E;
017677  PROCEDURE GetVideoDefault(paramBlock: DefVideoPtr);
017678      INLINE $205F,$A080;
017679  PROCEDURE SetVideoDefault(paramBlock: DefVideoPtr);
017680      INLINE $205F,$A081;
017681  PROCEDURE GetOSDefault(paramBlock: DefOSPtr);
017682      INLINE $205F,$A084;
017683  PROCEDURE SetOSDefault(paramBlock: DefOSPtr);
017684      INLINE $205F,$A083;
017685  PROCEDURE SetTimeout(count: INTEGER);
017686  PROCEDURE GetTimeout(VAR count: INTEGER);
017687
017688
017689  {$ENDC}    { UsingStart }
017690
017691  {$IFC NOT UsingIncludes}
017692      END.
017693  {$ENDC}
017694
017695
017696  ### END OF FILE Start.p
017697
```

```
017698
017699  ############################################################################
017700  ### FILE: Strings.p
017701  ############################################################################
017702
017703  {
017704  Created: Friday, October 20, 1989 at 3:26 PM
017705      Strings.p
017706      Pascal Interface to the Macintosh Libraries
017707
017708      Copyright Apple Computer, Inc.  1985-1989
017709      All rights reserved
017710  }
017711
017712
017713  {$IFC UNDEFINED UsingIncludes}
017714  {$SETC UsingIncludes := 0}
017715  {$ENDC}
017716
017717  {$IFC NOT UsingIncludes}
017718      UNIT Strings;
017719      INTERFACE
017720  {$ENDC}
017721
017722  {$IFC UNDEFINED UsingStrings}
017723  {$SETC UsingStrings := 1}
017724
017725  {$I+}
017726  {$SETC StringsIncludes := UsingIncludes}
017727  {$SETC UsingIncludes := 1}
017728  {$IFC UNDEFINED UsingTypes}
017729  {$I $$Shell(PInterfaces)Types.p}
017730  {$ENDC}
017731  {$SETC UsingIncludes := StringsIncludes}
017732
017733
017734  FUNCTION C2PStr(cString: UNIV Ptr): StringPtr;
017735  PROCEDURE C2PStrProc(cString: UNIV Ptr);
017736  FUNCTION P2CStr(pString: StringPtr): Ptr;
017737  PROCEDURE P2CStrProc(pString: StringPtr);
017738
017739  {$ENDC}    { UsingStrings }
017740
017741  {$IFC NOT UsingIncludes}
017742      END.
017743  {$ENDC}
017744
017745
017746  ### END OF FILE Strings.p
017747
```

```
017748
017749  ############################################################################
017750  ### FILE: SysEqu.p
017751  ############################################################################
017752
017753
017754  {
017755  Created: Friday, November 15, 1991 at 9:35 AM
017756   SysEqu.p
017757   Pascal Interface to the Macintosh Libraries
017758
017759    Copyright Apple Computer, Inc. 1985-1991
017760    All rights reserved
017761  }
017762
017763
017764  {$IFC UNDEFINED UsingIncludes}
017765  {$SETC UsingIncludes := 0}
017766  {$ENDC}
017767
017768  {$IFC NOT UsingIncludes}
017769   UNIT SysEqu;
017770   INTERFACE
017771  {$ENDC}
017772
017773  {$IFC UNDEFINED UsingSysEqu}
017774  {$SETC UsingSysEqu := 1}
017775
017776
017777  CONST
017778  PCDeskPat = $20B;               {[GLOBAL VAR]  desktop pat, top bit only! others are in use}
017779  HiKeyLast = $216;               {[GLOBAL VAR]  Same as KbdVars}
017780  KbdLast = $218;                 {[GLOBAL VAR]  Same as KbdVars+2}
017781  ExpandMem = $2B6;               {[GLOBAL VAR]  pointer to expanded memory block}
017782  SCSIBase = $0C00;               {[GLOBAL VAR]  (long) base address for SCSI chip read}
017783  SCSIDMA = $0C04;                {[GLOBAL VAR]  (long) base address for SCSI DMA}
017784  SCSIHsk = $0C08;                {[GLOBAL VAR]  (long) base address for SCSI handshake}
017785  SCSIGlobals = $0C0C;            {[GLOBAL VAR]  (long) ptr for SCSI mgr locals}
017786  RGBBlack = $0C10;               {[GLOBAL VAR]  (6 bytes) the black field for color}
017787  RGBWhite = $0C16;               {[GLOBAL VAR]  (6 bytes) the white field for color}
017788  RowBits = $0C20;                {[GLOBAL VAR]  (word) screen horizontal pixels}
017789  ColLines = $0C22;               {[GLOBAL VAR]  (word) screen vertical pixels}
017790  ScreenBytes = $0C24;            {[GLOBAL VAR]  (long) total screen bytes}
017791  NMIFlag = $0C2C;                {[GLOBAL VAR]  (byte) flag for NMI debounce}
017792  VidType = $0C2D;                {[GLOBAL VAR]  (byte) video board type ID}
017793  VidMode = $0C2E;                {[GLOBAL VAR]  (byte) video mode (4=4bit color)}
017794  SCSIPoll = $0C2F;               {[GLOBAL VAR]  (byte) poll for device zero only once.}
017795  SEVarBase = $0C30;              {[GLOBAL VAR] }
017796  MMUFlags = $0CB0;               {[GLOBAL VAR]  (byte) cleared to zero (reserved for future use)}
017797  MMUType = $0CB1;                {[GLOBAL VAR]  (byte) kind of MMU present}
017798  MMU32bit = $0CB2;               {[GLOBAL VAR]  (byte) boolean reflecting current machine MMU mode}
017799  MMUFluff = $0CB3;               {[GLOBAL VAR]  (byte) fluff byte forced by reducing MMUMode to MMU32bit.}
017800  MMUTbl = $0CB4;                 {[GLOBAL VAR]  (long) pointer to MMU Mapping table}
017801  MMUTblSize = $0CB8;             {[GLOBAL VAR]  (long) size of the MMU mapping table}
017802  SInfoPtr = $0CBC;               {[GLOBAL VAR]  (long) pointer to Slot manager information}
017803  ASCBase = $0CC0;                {[GLOBAL VAR]  (long) pointer to Sound Chip}
```

```
017804  SMGlobals = $0CC4;              { (long) pointer to Sound Manager Globals}
017805  TheGDevice = $0CC8;             {[GLOBAL VAR]  (long) the current graphics device}
017806  CQDGlobals = $0CCC;             { (long) quickDraw global extensions}
017807  ADBBase = $0CF8;                {[GLOBAL VAR]  (long) pointer to Front Desk Buss Variables}
017808  WarmStart = $0CFC;              {[GLOBAL VAR]  (long) flag to indicate it is a warm start}
017809  TimeDBRA = $0D00;               {[GLOBAL VAR]  (word) number of iterations of DBRA per millisecond}
017810  TimeSCCDB = $0D02;              {[GLOBAL VAR]  (word) number of iter's of SCC access & DBRA.}
017811  SlotQDT = $0D04;                {[GLOBAL VAR]  ptr to slot queue table}
017812  SlotPrTbl = $0D08;              {[GLOBAL VAR]  ptr to slot priority table}
017813  SlotVBLQ = $0D0C;               {[GLOBAL VAR]  ptr to slot VBL queue table}
017814  ScrnVBLPtr = $0D10;             {[GLOBAL VAR]  save for ptr to main screen VBL queue}
017815  SlotTICKS = $0D14;              {[GLOBAL VAR]  ptr to slot tickcount table}
017816  TableSeed = $0D20;              {[GLOBAL VAR]  (long) seed value for color table ID's}
017817  SRsrcTblPtr = $0D24;            {[GLOBAL VAR]  (long) pointer to slot resource table.}
017818  JVBLTask = $0D28;               {[GLOBAL VAR]  vector to slot VBL task interrupt handler}
017819  WMgrCPort = $0D2C;              {[GLOBAL VAR]  window manager color port }
017820  VertRRate = $0D30;              {[GLOBAL VAR]  (word) Vertical refresh rate for start manager. }
017821  ChunkyDepth = $0D60;            {[GLOBAL VAR]  depth of the pixels}
017822  CrsrPtr = $0D62;                {[GLOBAL VAR]  pointer to cursor save area}
017823  PortList = $0D66;               {[GLOBAL VAR]  list of grafports}
017824  MickeyBytes = $0D6A;            {[GLOBAL VAR]  long pointer to cursor stuff}
017825  QDErrLM = $0D6E;                {[GLOBAL VAR] QDErr has name conflict w/ type. QuickDraw error code [word]}
017826  VIA2DT = $0D70;                 {[GLOBAL VAR]  32 bytes for VIA2 dispatch table for NuMac}
017827  SInitFlags = $0D90;             {[GLOBAL VAR]  StartInit.a flags [word]}
017828  DTQueue = $0D92;                {[GLOBAL VAR]  (10 bytes) deferred task queue header}
017829  DTQFlags = $0D92;               {[GLOBAL VAR]  flag word for DTQueue}
017830  DTskQHdr = $0D94;               {[GLOBAL VAR]  ptr to head of queue}
017831  DTskQTail = $0D98;              {[GLOBAL VAR]  ptr to tail of queue}
017832  JDTInstall = $0D9C;             {[GLOBAL VAR]  (long) ptr to deferred task install routine}
017833  HiliteRGB = $0DA0;              {[GLOBAL VAR]  6 bytes: rgb of hilite color}
017834  TimeSCSIDB = $0B24;             {[GLOBAL VAR]  (word) number of iter's of SCSI access & DBRA}
017835  DSCtrAdj = $0DA8;               {[GLOBAL VAR]  (long) Center adjust for DS rect.}
017836  IconTLAddr = $0DAC;             {[GLOBAL VAR]  (long) pointer to where start icons are to be put.}
017837  VideoInfoOK = $0DB0;            {[GLOBAL VAR]  (long) Signals to CritErr that the Video card is ok}
017838  EndSRTPtr = $0DB4;              {[GLOBAL VAR]  (long) Pointer to the end of the Slot Resource Table (Not the SRT buffer).}
017839  SDMJmpTblPtr = $0DB8;           {[GLOBAL VAR]  (long) Pointer to the SDM jump table}
017840  JSwapMMU = $0DBC;               {[GLOBAL VAR]  (long) jump vector to SwapMMU routine}
017841  SdmBusErr = $0DC0;              {[GLOBAL VAR]  (long) Pointer to the SDM busErr handler}
017842  LastTxGDevice = $0DC4;          {[GLOBAL VAR]  (long) copy of TheGDevice set up for fast text measure}
017843  NewCrsrJTbl = $88C;             {[GLOBAL VAR]  location of new crsr jump vectors}
017844  JAllocCrsr = $88C;              {[GLOBAL VAR]  (long) vector to routine that allocates cursor}
017845  JSetCCrsr = $890;               {[GLOBAL VAR]  (long) vector to routine that sets color cursor}
017846  JOpcodeProc = $894;             {[GLOBAL VAR]  (long) vector to process new picture opcodes}
017847  CrsrBase = $898;                {[GLOBAL VAR]  (long) scrnBase for cursor}
017848  CrsrDevice = $89C;              {[GLOBAL VAR]  (long) current cursor device}
017849  SrcDevice = $8A0;               {[GLOBAL VAR]  (LONG) Src device for Stretchbits}
017850  MainDevice = $8A4;              {[GLOBAL VAR]  (long) the main screen device}
017851  DeviceList = $8A8;              {[GLOBAL VAR]  (long) list of display devices}
017852  CrsrRow = $8AC;                 {[GLOBAL VAR]  (word) rowbytes for current cursor screen}
017853  QDColors = $8B0;                {[GLOBAL VAR]  (long) handle to default colors}
017854  HiliteMode = $938;              {[GLOBAL VAR]  used for color highlighting}
017855  BusErrVct = $08;                {[GLOBAL VAR]  bus error vector}
017856  RestProc = $A8C;                {[GLOBAL VAR]  Resume procedure f InitDialogs [pointer]}
017857  ROM85 = $28E;                   {[GLOBAL VAR]  (word) actually high bit - 0 for ROM vers $75 (sic) and later}
017858  ROMMapHndl = $B06;              {[GLOBAL VAR]  (long) handle of ROM resource map}
017859  ScrVRes = $102;                 {[GLOBAL VAR] Pixels per inch vertically (word)
```

```
017860     screen vertical dots/inch [word]}
017861  ScrHRes = $104;                 {[GLOBAL VAR] Pixels per inch horizontally (word)
017862     screen horizontal dots/inch [word]}
017863  ScrnBase = $824;                {[GLOBAL VAR] Address of main screen buffer
017864     Screen Base [pointer]}
017865  ScreenRow = $106;               {[GLOBAL VAR]  rowBytes of screen [word]}
017866  MBTicks = $16E;                 {[GLOBAL VAR]  tick count @ last mouse button [long]}
017867  JKybdTask = $21A;               {[GLOBAL VAR]  keyboard VBL task hook [pointer]}
017868  KeyLast = $184;                 {[GLOBAL VAR]  ASCII for last valid keycode [word]}
017869  KeyTime = $186;                 {[GLOBAL VAR]  tickcount when KEYLAST was rec'd [long]}
017870  KeyRepTime = $18A;              {[GLOBAL VAR]  tickcount when key was last repeated [long]}
017871  SPConfig = $1FB;                {[GLOBAL VAR] Use types for serial ports (byte)
017872     config bits: 4-7 A, 0-3 B (see use type below)}
017873  SPPortA = $1FC;                 {[GLOBAL VAR] Modem port configuration (word)
017874     SCC port A configuration [word]}
017875  SPPortB = $1FE;                 {[GLOBAL VAR] Printer port configuration (word)
017876     SCC port B configuration [word]}
017877  SCCRd = $1D8;                   {[GLOBAL VAR] SCC read base address
017878     SCC base read address [pointer]}
017879  SCCWr = $1DC;                   {[GLOBAL VAR] SCC write base address
017880     SCC base write address [pointer]}
017881  DoubleTime = $2F0;              {[GLOBAL VAR] Double-click interval in ticks (long)
017882     double click ticks [long]}
017883  CaretTime = $2F4;               {[GLOBAL VAR] Caret-blink interval in ticks (long)
017884     caret blink ticks [long]}
017885  KeyThresh = $18E;               {[GLOBAL VAR] Auto-key threshold (word)
017886     threshold for key repeat [word]}
017887  KeyRepThresh = $190;            {[GLOBAL VAR] Auto-key rate (word)
017888     key repeat speed [word]}
017889  SdVolume = $260;                {[GLOBAL VAR] Current speaker volume (byte:  low-order three bits only)
017890     Global volume(sound) control [byte]}
017891  Ticks = $16A;                   {[GLOBAL VAR] Current number of ticks since system startup (long)
017892     Tick count, time since boot [unsigned long]}
017893  TimeLM = $20C;                  {[GLOBAL VAR] Time has name conflict w/ type. Clock time (extrapolated) [long]}
017894  MonkeyLives = $100;             {[GLOBAL VAR]  monkey lives if >= 0 [word]}
017895  SEvtEnb = $15C;                 {[GLOBAL VAR] 0 if SystemEvent should return FALSE (byte)
017896     enable SysEvent calls from GNE [byte]}
017897  JournalFlag = $8DE;             {[GLOBAL VAR] Journaling mode (word)
017898     journaling state [word]}
017899  JournalRef = $8E8;              {[GLOBAL VAR] Reference number of journaling device driver (word)
017900     Journalling driver's refnum [word]}
017901  BufPtr = $10C;                  {[GLOBAL VAR] Address of end of jump table
017902     top of application memory [pointer]}
017903  StkLowPt = $110;                {[GLOBAL VAR]  Lowest stack as measured in VBL task [pointer]}
017904  TheZone = $118;                 {[GLOBAL VAR] Address of current heap zone
017905     current heap zone [pointer]}
017906  ApplLimit = $130;               {[GLOBAL VAR] Application heap limit
017907     application limit [pointer]}
017908  SysZone = $2A6;                 {[GLOBAL VAR] Address of system heap zone
017909     system heap zone [pointer]}
017910  ApplZone = $2AA;                {[GLOBAL VAR] Address of application heap zone
017911     application heap zone [pointer]}
017912  HeapEnd = $114;                 {[GLOBAL VAR] Address of end of application heap zone
017913     end of heap [pointer]}
017914  HiHeapMark = $BAE;              {[GLOBAL VAR]  (long) highest address used by a zone below sp<01Nov85 JTC>}
017915  MemErr = $220;                  {[GLOBAL VAR]  last memory manager error [word]}
```

```
017916  UTableBase = $11C;              {[GLOBAL VAR] Base address of unit table
017917     unit I/O table [pointer]}
017918  UnitNtryCnt = $1D2;             {[GLOBAL VAR]  count of entries in unit table [word]}
017919  JFetch = $8F4;                  {[GLOBAL VAR] Jump vector for Fetch function
017920     fetch a byte routine for drivers [pointer]}
017921  JStash = $8F8;                  {[GLOBAL VAR] Jump vector for Stash function
017922     stash a byte routine for drivers [pointer]}
017923  JIODone = $8FC;                 {[GLOBAL VAR] Jump vector for IODone function
017924     IODone entry location [pointer]}
017925  DrvQHdr = $308;                 {[GLOBAL VAR] Drive queue header (10 bytes)
017926     queue header of drives in system [10 bytes]}
017927  BootDrive = $210;               {[GLOBAL VAR]  drive number of boot drive [word]}
017928  EjectNotify = $338;             {[GLOBAL VAR]  eject notify procedure [pointer]}
017929  IAZNotify = $33C;               {[GLOBAL VAR]  world swaps notify procedure [pointer]}
017930  SFSaveDisk = $214;              {[GLOBAL VAR] Negative of volume reference number used by Standard File Package (word)
017931     last vRefNum seen by standard file [word]}
017932  CurDirStore = $398;             {[GLOBAL VAR]  save dir across calls to Standard File [long]}
017933  OneOne = $A02;                  {[GLOBAL VAR] $00010001
017934     constant $00010001 [long]}
017935  MinusOne = $A06;                {[GLOBAL VAR] $FFFFFFFF
017936     constant $FFFFFFFF [long]}
017937  Lo3Bytes = $31A;                {[GLOBAL VAR] $00FFFFFF
017938     constant $00FFFFFF [long]}
017939  ROMBase = $2AE;                 {[GLOBAL VAR] Base address of ROM
017940     ROM base address [pointer]}
017941  RAMBase = $2B2;                 {[GLOBAL VAR] Trap dispatch table's base address for routines in RAM
017942     RAM base address [pointer]}
017943  SysVersion = $15A;              {[GLOBAL VAR]  version # of RAM-based system [word]}
017944  RndSeed = $156;                 {[GLOBAL VAR] Random number seed (long)
017945     random seed/number [long]}
017946  Scratch20 = $1E4;               {[GLOBAL VAR] 20-byte scratch area
017947     scratch [20 bytes]}
017948  Scratch8 = $9FA;                {[GLOBAL VAR] 8-byte scratch area
017949     scratch [8 bytes]}
017950  ToolScratch = $9CE;             {[GLOBAL VAR] 8-byte scratch area
017951     scratch [8 bytes]}
017952  ApplScratch = $A78;             {[GLOBAL VAR] 12-byte application scratch area
017953     scratch [12 bytes]}
017954  ScrapSize = $960;               {[GLOBAL VAR] Size in bytes of desk scrap (long)
017955     scrap length [long]}
017956  ScrapHandle = $964;             {[GLOBAL VAR] Handle to desk scrap in memory
017957     memory scrap [handle]}
017958  ScrapCount = $968;              {[GLOBAL VAR] Count changed by ZeroScrap (word)
017959     validation byte [word]}
017960  ScrapState = $96A;              {[GLOBAL VAR] Tells where desk scrap is (word)
017961     scrap state [word]}
017962  ScrapName = $96C;               {[GLOBAL VAR] Pointer to scrap file name (preceded by length byte)
017963     pointer to scrap name [pointer]}
017964  IntlSpec = $BA0;                {[GLOBAL VAR]  (long) - ptr to extra Intl data }
017965  SwitcherTPtr = $286;            {[GLOBAL VAR]  Switcher's switch table }
017966  CPUFlag = $12F;                 {[GLOBAL VAR]  $00=68000, $01=68010, $02=68020 (old ROM inits to $00)}
017967  VIA = $1D4;                     {[GLOBAL VAR] VIA base address
017968     VIA base address [pointer]}
017969  IWM = $1E0;                     {[GLOBAL VAR]  IWM base address [pointer]}
017970  Lvl1DT = $192;                  {[GLOBAL VAR] Level-1 secondary interrupt vector table (32 bytes)
017971     Interrupt level 1 dispatch table [32 bytes]}
```

```
017972  Lvl2DT = $1B2;                  {[GLOBAL VAR] Level-2 secondary interrupt vector table (32 bytes)
017973     Interrupt level 2 dispatch table [32 bytes]}
017974  ExtStsDT = $2BE;                {[GLOBAL VAR] External/status interrupt vector table (16 bytes)
017975     SCC ext/sts secondary dispatch table [16 bytes]}
017976  SPValid = $1F8;                 {[GLOBAL VAR] Validity status (byte)
017977     validation field ($A7) [byte]}
017978  SPATalkA = $1F9;                {[GLOBAL VAR] AppleTalk node ID hint for modem port (byte)
017979     AppleTalk node number hint for port A}
017980  SPATalkB = $1FA;                {[GLOBAL VAR] AppleTalk node ID hint for printer port (byte)
017981     AppleTalk node number hint for port B}
017982  SPAlarm = $200;                 {[GLOBAL VAR] Alarm setting (long)
017983     alarm time [long]}
017984  SPFont = $204;                  {[GLOBAL VAR] Application font number minus 1 (word)
017985     default application font number minus 1 [word]}
017986  SPKbd = $206;                   {[GLOBAL VAR] Auto-key threshold and rate (byte)
017987     kbd repeat thresh in 4/60ths [2 4-bit]}
017988  SPPrint = $207;                 {[GLOBAL VAR] Printer connection (byte)
017989     print stuff [byte]}
017990  SPVolCtl = $208;                {[GLOBAL VAR] Speaker volume setting in parameter RAM (byte)
017991     volume control [byte]}
017992  SPClikCaret = $209;             {[GLOBAL VAR] Double-click and caret-blink times (byte)
017993     double click/caret time in 4/60ths[2 4-bit]}
017994  SPMisc1 = $20A;                 {[GLOBAL VAR]  miscellaneous [1 byte]}
017995  SPMisc2 = $20B;                 {[GLOBAL VAR] Mouse scaling, system startup disk, menu blink (byte)
017996     miscellaneous [1 byte]}
017997  GetParam = $1E4;                {[GLOBAL VAR]  system parameter scratch [20 bytes]}
017998  SysParam = $1F8;                {[GLOBAL VAR] Low-memory copy of parameter RAM (20 bytes)
017999     system parameter memory [20 bytes]}
018000  CrsrThresh = $8EC;              {[GLOBAL VAR] Mouse-scaling threshold (word)
018001     delta threshold for mouse scaling [word]}
018002  JCrsrTask = $8EE;               {[GLOBAL VAR]  address of CrsrVBLTask [long]}
018003  MTemp = $828;                   {[GLOBAL VAR]  Low-level interrupt mouse location [long]}
018004  RawMouse = $82C;                {[GLOBAL VAR]  un-jerked mouse coordinates [long]}
018005  CrsrRect = $83C;                {[GLOBAL VAR]  Cursor hit rectangle [8 bytes]}
018006  TheCrsr = $844;                 {[GLOBAL VAR]  Cursor data, mask & hotspot [68 bytes]}
018007  CrsrAddr = $888;                {[GLOBAL VAR]  Address of data under cursor [long]}
018008  CrsrSave = $88C;                {[GLOBAL VAR]  data under the cursor [64 bytes]}
018009  CrsrVis = $8CC;                 {[GLOBAL VAR]  Cursor visible? [byte]}
018010  CrsrBusy = $8CD;                {[GLOBAL VAR]  Cursor locked out? [byte]}
018011  CrsrNew = $8CE;                 {[GLOBAL VAR]  Cursor changed? [byte]}
018012  CrsrState = $8D0;               {[GLOBAL VAR]  Cursor nesting level [word]}
018013  CrsrObscure = $8D2;             {[GLOBAL VAR]  Cursor obscure semaphore [byte]}
018014  KbdVars = $216;                 {[GLOBAL VAR]  Keyboard manager variables [4 bytes]}
018015  KbdType = $21E;                 {[GLOBAL VAR]  keyboard model number [byte]}
018016  MBState = $172;                 {[GLOBAL VAR]  current mouse button state [byte]}
018017  KeyMapLM = $174;                {[GLOBAL VAR] KeyMap has name conflict w/ type. Bitmap of the keyboard [4 longs]}
018018  KeypadMap = $17C;               {[GLOBAL VAR]  bitmap for numeric pad-18bits [long]}
018019  Key1Trans = $29E;               {[GLOBAL VAR]  keyboard translator procedure [pointer]}
018020  Key2Trans = $2A2;               {[GLOBAL VAR]  numeric keypad translator procedure [pointer]}
018021  JGNEFilter = $29A;              {[GLOBAL VAR]  GetNextEvent filter proc [pointer]}
018022  KeyMVars = $B04;                {[GLOBAL VAR]  (word) for ROM KEYM proc state}
018023  Mouse = $830;                   {[GLOBAL VAR]  processed mouse coordinate [long]}
018024  CrsrPin = $834;                 {[GLOBAL VAR]  cursor pinning rectangle [8 bytes]}
018025  CrsrCouple = $8CF;              {[GLOBAL VAR]  cursor coupled to mouse? [byte]}
018026  CrsrScale = $8D3;               {[GLOBAL VAR]  cursor scaled? [byte]}
018027  MouseMask = $8D6;               {[GLOBAL VAR]  V-H mask for ANDing with mouse [long]}
```

```
018028  MouseOffset = $8DA;             {[GLOBAL VAR]  V-H offset for adding after ANDing [long]}
018029  AlarmState = $21F;              {[GLOBAL VAR]  Bit7=parity, Bit6=beeped, Bit0=enable [byte]}
018030  VBLQueue = $160;                {[GLOBAL VAR] Vertical retrace queue header (10 bytes)
018031     VBL queue header [10 bytes]}
018032  SysEvtMask = $144;              {[GLOBAL VAR] System event mask (word)
018033     system event mask [word]}
018034  SysEvtBuf = $146;               {[GLOBAL VAR]  system event queue element buffer [pointer]}
018035  EventQueue = $14A;              {[GLOBAL VAR] Event queue header (10 bytes)
018036     event queue header [10 bytes]}
018037  EvtBufCnt = $154;               {[GLOBAL VAR]  max number of events in SysEvtBuf - 1 [word]}
018038  GZRootHnd = $328;               {[GLOBAL VAR] Handle to relocatable block not to be moved by grow zone function
018039     root handle for GrowZone [handle]}
018040  GZRootPtr = $32C;               {[GLOBAL VAR]  root pointer for GrowZone [pointer]}
018041  GZMoveHnd = $330;               {[GLOBAL VAR]  moving handle for GrowZone [handle]}
018042  MemTop = $108;                  {[GLOBAL VAR] Address of end of RAM (on Macintosh XL, end of RAM available to applications)
018043     top of memory [pointer]}
018044  MmInOK = $12E;                  {[GLOBAL VAR]  initial memory mgr checks ok? [byte]}
018045  HpChk = $316;                   {[GLOBAL VAR]  heap check RAM code [pointer]}
018046  MaskBC = $31A;                  {[GLOBAL VAR]  Memory Manager Byte Count Mask [long]}
018047  MaskHandle = $31A;              {[GLOBAL VAR]  Memory Manager Handle Mask [long]}
018048  MaskPtr = $31A;                 {[GLOBAL VAR]  Memory Manager Pointer Mask [long]}
018049  MinStack = $31E;                {[GLOBAL VAR] Minimum space allotment for stack (long)
018050     min stack size used in InitApplZone [long]}
018051  DefltStack = $322;              {[GLOBAL VAR] Default space allotment for stack (long)
018052     default size of stack [long]}
018053  MMDefFlags = $326;              {[GLOBAL VAR]  default zone flags [word]}
018054  DSAlertTab = $2BA;              {[GLOBAL VAR] Pointer to system error alert table in use
018055     system error alerts [pointer]}
018056  DSAlertRect = $3F8;             {[GLOBAL VAR] Rectangle enclosing system error alert (8 bytes)
018057     rectangle for disk-switch alert [8 bytes]}
018058  DSDrawProc = $334;              {[GLOBAL VAR]  alternate syserror draw procedure [pointer]}
018059  DSWndUpdate = $15D;             {[GLOBAL VAR]  GNE not to paintBehind DS AlertRect? [byte]}
018060  WWExist = $8F2;                 {[GLOBAL VAR]  window manager initialized? [byte]}
018061  QDExist = $8F3;                 {[GLOBAL VAR]  quickdraw is initialized [byte]}
018062  ResumeProc = $A8C;              {[GLOBAL VAR] Address of resume procedure
018063     Resume procedure from InitDialogs [pointer]}
018064  DSErrCode = $AF0;               {[GLOBAL VAR] Current system error ID (word)
018065     last system error alert ID}
018066  IntFlag = $15F;                 {[GLOBAL VAR]  reduce interrupt disable time when bit 7 = 0}
018067  SerialVars = $2D0;              {[GLOBAL VAR]  async driver variables [16 bytes]}
018068  ABusVars = $2D8;                {[GLOBAL VAR] Pointer to AppleTalk variables
018069    ;Pointer to AppleTalk local variables}
018070  ABusDCE = $2DC;                 {[GLOBAL VAR] ;Pointer to AppleTalk DCE}
018071  PortAUse = $290;                {[GLOBAL VAR]  bit 7: 1 = not in use, 0 = in use}
018072  PortBUse = $291;                {[GLOBAL VAR] Current availability of serial port B (byte)
018073     port B use, same format as PortAUse}
018074  SCCASts = $2CE;                 {[GLOBAL VAR]  SCC read reg 0 last ext/sts rupt - A [byte]}
018075  SCCBSts = $2CF;                 {[GLOBAL VAR]  SCC read reg 0 last ext/sts rupt - B [byte]}
018076  DskErr = $142;                  {[GLOBAL VAR]  disk routine result code [word]}
018077  PWMBuf2 = $312;                 {[GLOBAL VAR]  PWM buffer 1 (or 2 if sound) [pointer]}
018078  SoundPtr = $262;                {[GLOBAL VAR] Pointer to four-tone record
018079     4VE sound definition table [pointer]}
018080  SoundBase = $266;               {[GLOBAL VAR] Pointer to free-form synthesizer buffer
018081     sound bitMap [pointer]}
018082  SoundVBL = $26A;                {[GLOBAL VAR]  vertical retrace control element [16 bytes]}
018083  SoundDCE = $27A;                {[GLOBAL VAR]  sound driver DCE [pointer]}
```

```
018084  SoundActive = $27E;             {[GLOBAL VAR]  sound is active? [byte]}
018085  SoundLevel = $27F;              {[GLOBAL VAR] Amplitude in 740-byte buffer (byte)
018086     current level in buffer [byte]}
018087  CurPitch = $280;                {[GLOBAL VAR] Value of count in square-wave synthesizer buffer (word)
018088     current pitch value [word]}
018089  DskVerify = $12C;               {[GLOBAL VAR]  used by 3.5 disk driver for read/verify [byte]}
018090  TagData = $2FA;                 {[GLOBAL VAR]  sector tag info for disk drivers [14 bytes]}
018091  BufTgFNum = $2FC;               {[GLOBAL VAR] File tags buffer:  file number (long)
018092     file number [long]}
018093  BufTgFFlg = $300;               {[GLOBAL VAR] File tags buffer:  flags (word:  bit 1=1 if resource fork)
018094     flags [word]}
018095  BufTgFBkNum = $302;             {[GLOBAL VAR] File tags buffer:  logical block number (word)
018096     logical block number [word]}
018097  BufTgDate = $304;               {[GLOBAL VAR] File tags buffer:  date and time of last modification (long)
018098     time stamp [word]}
018099  ScrDmpEnb = $2F8;               {[GLOBAL VAR] 0 if GetNextEvent shouldn't process Command-Shift-number combinations (byte)
018100     screen dump enabled? [byte]}
018101  ScrDmpType = $2F9;              {[GLOBAL VAR]  FF dumps screen, FE dumps front window [byte]}
018102  ScrapVars = $960;               {[GLOBAL VAR]  scrap manager variables [32 bytes]}
018103  ScrapInfo = $960;               {[GLOBAL VAR]  scrap length [long]}
018104  ScrapEnd = $980;                {[GLOBAL VAR]  end of scrap vars}
018105  ScrapTag = $970;                {[GLOBAL VAR]  scrap file name [STRING[15]]}
018106  LaunchFlag = $902;              {[GLOBAL VAR]  from launch or chain [byte]}
018107  SaveSegHandle = $930;           {[GLOBAL VAR]  seg 0 handle [handle]}
018108  CurJTOffset = $934;             {[GLOBAL VAR] Offset to jump table from location pointed to by A5 (word)
018109     current jump table offset [word]}
018110  CurPageOption = $936;           {[GLOBAL VAR] Sound/screen buffer configuration passed to Chain or Launch (word)
018111     current page 2 configuration [word]}
018112  LoaderPBlock = $93A;            {[GLOBAL VAR]  param block for ExitToShell [10 bytes]}
018113  CurApRefNum = $900;             {[GLOBAL VAR] Reference number of current application's resource file (word)
018114     refNum of application's resFile [word]}
018115  CurrentA5 = $904;               {[GLOBAL VAR] Address of boundary between application globals and application parameters
018116     current value of A5 [pointer]}
018117  CurStackBase = $908;            {[GLOBAL VAR] Address of base of stack; start of application globals
018118     current stack base [pointer]}
018119  CurApName = $910;               {[GLOBAL VAR] Name of current application (length byte followed by up to 31 characters)
018120     name of application [STRING[31]]}
018121  LoadTrap = $12D;                {[GLOBAL VAR]  trap before launch? [byte]}
018122  SegHiEnable = $BB2;             {[GLOBAL VAR]  (byte) 0 to disable MoveHHi in LoadSeg}
018123
018124  { Window Manager Globals }
018125  WindowList = $9D6;              {[GLOBAL VAR] Pointer to first window in window list; 0 if using events but not windows
018126    Z-ordered linked list of windows [pointer]}
018127  PaintWhite = $9DC;              {[GLOBAL VAR] Flag for whether to paint window white before update event (word)
018128    erase newly drawn windows? [word]}
018129  WMgrPort = $9DE;                {[GLOBAL VAR] Pointer to Window Manager port
018130    window manager's grafport [pointer]}
018131  GrayRgn = $9EE;                 {[GLOBAL VAR] Handle to region drawn as desktop
018132    rounded gray desk region [handle]}
018133  CurActivate = $A64;             {[GLOBAL VAR] Pointer to window to receive activate event
018134    window slated for activate event [pointer]}
018135  CurDeactive = $A68;             {[GLOBAL VAR] Pointer to window to receive deactivate event
018136    window slated for deactivate event [pointer]}
018137  DragHook = $9F6;                {[GLOBAL VAR] Address of procedure to execute during TrackGoAway, DragWindow, GrowWindow, DragGrayRgn, TrackContro
018138    user hook during dragging [pointer]}
018139  DeskPattern = $A3C;             {[GLOBAL VAR] Pattern with which desktop is painted (8 bytes)
```

```
018140    desk pattern [8 bytes]}
018141  DeskHook = $A6C;                {[GLOBAL VAR] Address of procedure for painting desktop or responding to clicks on desktop
018142    hook for painting the desk [pointer]}
018143  GhostWindow = $A84;             {[GLOBAL VAR] Pointer to window never to be considered frontmost
018144    window hidden from FrontWindow [pointer]}
018145
018146  { Text Edit Globals }
018147  TEDoText = $A70;                {[GLOBAL VAR] Address of TextEdit multi-purpose routine
018148    textEdit doText proc hook [pointer]}
018149  TERecal = $A74;                 {[GLOBAL VAR] Address of routine to recalculate line starts for TextEdit
018150    textEdit recalText proc hook [pointer]}
018151  TEScrpLength = $AB0;            {[GLOBAL VAR] Size in bytes of TextEdit scrap (long)
018152    textEdit Scrap Length [word]}
018153  TEScrpHandle = $AB4;            {[GLOBAL VAR] Handle to TextEdit scrap
018154    textEdit Scrap [handle]}
018155  TEWdBreak = $AF6;               {[GLOBAL VAR] default word break routine [pointer]}
018156  WordRedraw = $BA5;              {[GLOBAL VAR] (byte) - used by TextEdit RecalDraw}
018157  TESysJust = $BAC;               {[GLOBAL VAR] (word) system justification (intl. textEdit)}
018158
018159  { Resource Manager Globals }
018160  TopMapHndl = $A50;              {[GLOBAL VAR] Handle to resource map of most recently opened resource file
018161    topmost map in list [handle]}
018162  SysMapHndl = $A54;              {[GLOBAL VAR] Handle to map of system resource file
018163    system map [handle]}
018164  SysMap = $A58;                  {[GLOBAL VAR] Reference number of system resource file (word)
018165    reference number of system map [word]}
018166  CurMap = $A5A;                  {[GLOBAL VAR] Reference number of current resource file (word)
018167    reference number of current map [word]}
018168  ResReadOnly = $A5C;             {[GLOBAL VAR] Read only flag [word]}
018169  ResLoad = $A5E;                 {[GLOBAL VAR] Current SetResLoad state (word)
018170    Auto-load feature [word]}
018171  ResErr = $A60;                  {[GLOBAL VAR] Current value of ResError (word)
018172    Resource error code [word]}
018173  ResErrProc = $AF2;              {[GLOBAL VAR] Address of resource error procedure
018174    Resource error procedure [pointer]}
018175  SysResName = $AD8;              {[GLOBAL VAR] Name of system resource file (length byte followed by up to 19 characters)
018176    Name of system resource file [STRING[19]]}
018177  RomMapInsert = $B9E;            {[GLOBAL VAR] (byte) determines if we should link in map}
018178  TmpResLoad = $B9F;              {[GLOBAL VAR] second byte is temporary ResLoad value.}
018179
018180  { Menu Mgr globals }
018181  MBarHeight = $BAA;              {[GLOBAL VAR] height of the menu bar}
018182
018183  { CommToolbox Global }
018184  CommToolboxGlobals = $0BB4;     {[GLOBAL VAR] pointer to CommToolbox globals }
018185
018186
018187  {$ENDC} { UsingSysEqu }
018188
018189  {$IFC NOT UsingIncludes}
018190   END.
018191  {$ENDC}
018192
018193
018194  ### END OF FILE SysEqu.p
018195
```

```
018196
018197  ############################################################################
018198  ### FILE: Terminals.p
018199  ############################################################################
018200
018201
018202  {
018203  Created: Thursday, September 12, 1991 at 10:33 AM
018204   Terminals.p
018205   Pascal Interface to the Macintosh Libraries
018206
018207    Copyright Apple Computer, Inc. 1988-1991
018208    All rights reserved
018209  }
018210
018211
018212  {$IFC UNDEFINED UsingIncludes}
018213  {$SETC UsingIncludes := 0}
018214  {$ENDC}
018215
018216  {$IFC NOT UsingIncludes}
018217   UNIT Terminals;
018218   INTERFACE
018219  {$ENDC}
018220
018221  {$IFC UNDEFINED UsingTerminals}
018222  {$SETC UsingTerminals := 1}
018223
018224  {$I+}
018225  {$SETC TerminalsIncludes := UsingIncludes}
018226  {$SETC UsingIncludes := 1}
018227  {$IFC UNDEFINED UsingDialogs}
018228  {$I $$Shell(PInterfaces)Dialogs.p}
018229  {$ENDC}
018230  {$IFC UNDEFINED UsingCTBUtilities}
018231  {$I $$Shell(PInterfaces)CTBUtilities.p}
018232  {$ENDC}
018233  {$IFC UNDEFINED UsingConnections}
018234  {$I $$Shell(PInterfaces)Connections.p}
018235  {$ENDC}
018236  {$SETC UsingIncludes := TerminalsIncludes}
018237
018238  CONST
018239
018240  { current Terminal Manager version }
018241  curTMVersion = 2;
018242
018243  { current Terminal Manager Environment Record version }
018244  curTermEnvRecVers = 0;
018245
018246  { error codes    }
018247  tmGenericError = -1;
018248  tmNoErr = 0;
018249  tmNotSent = 1;
018250  tmEnvironsChanged = 2;
018251  tmNotSupported = 7;
```

```
018252  tmNoTools = 8;
018253
018254  TYPE
018255  TMErr = OSErr;
018256
018257  CONST
018258
018259  { TMFlags }
018260  tmInvisible = $00000001;
018261  tmSaveBeforeClear = $00000002;
018262  tmNoMenus = $00000004;
018263  tmAutoScroll = $00000008;
018264  tmConfigChanged = $00000010;
018265
018266  TYPE
018267  {  TMFlags   }
018268  TMFlags = LONGINT;
018269
018270
018271  CONST
018272
018273  { TMSelTypes & TMSearchTypes }
018274  selTextNormal = $0001;
018275  selTextBoxed = $0002;
018276  selGraphicsMarquee = $0004;
018277  selGraphicsLasso = $0008;
018278  tmSearchNoDiacrit = $0100;  {These are only for TMSearchTypes}
018279  tmSearchNoCase = $0200;     {These are only for TMSearchTypes}
018280
018281  TYPE
018282  {  TMSelTypes & TMSearchTypes  }
018283  TMSearchTypes = INTEGER;
018284
018285
018286  TMSelTypes = INTEGER;
018287
018288  CONST
018289
018290  { TMCursorTypes }
018291  cursorText = 1;
018292  cursorGraphics = 2;
018293
018294  TYPE
018295  TMCursorTypes = INTEGER;
018296
018297
018298  CONST
018299
018300  { TMTermTypes }
018301  tmTextTerminal = $0001;
018302  tmGraphicsTerminal = $0002;
018303
018304  TYPE
018305  {  TMTermTypes   }
018306  TMTermTypes = INTEGER;
018307
```

```
018308
018309  TermDataBlockPtr = ^TermDataBlock;
018310  TermDataBlockH = ^TermDataBlockPtr;
018311  TermDataBlock = RECORD
018312   flags: TMTermTypes;
018313   theData: Handle;
018314   auxData: Handle;
018315   reserved: LONGINT;
018316   END;
018317
018318  TermEnvironPtr = ^TermEnvironRec;
018319  TermEnvironRec = RECORD
018320   version: INTEGER;
018321   termType: TMTermTypes;
018322   textRows: INTEGER;
018323   textCols: INTEGER;
018324   cellSize: Point;
018325   graphicSize: Rect;
018326   slop: Point;
018327   auxSpace: Rect;
018328   END;
018329
018330  TMSelection = RECORD
018331   CASE INTEGER OF
018332   1: (selRect: Rect);
018333   2: (selRgnHandle: RgnHandle;
018334   filler: LONGINT);
018335   END;
018336
018337  {    TMTermTypes     }
018338  TermPtr = ^TermRecord;
018339  TermHandle = ^TermPtr;
018340  TermRecord = RECORD
018341   procID: INTEGER;
018342   flags: TMFlags;
018343   errCode: TMErr;
018344   refCon: LONGINT;
018345   userData: LONGINT;
018346   defProc: ProcPtr;
018347   config: Ptr;
018348   oldConfig: Ptr;
018349   environsProc: ProcPtr;
018350   reserved1: LONGINT;
018351   reserved2: LONGINT;
018352   tmPrivate: Ptr;
018353   sendProc: ProcPtr;
018354   breakProc: ProcPtr;
018355   cacheProc: ProcPtr;
018356   clikLoop: ProcPtr;
018357   owner: WindowPtr;
018358   termRect: Rect;
018359   viewRect: Rect;
018360   visRect: Rect;
018361   lastIdle: LONGINT;
018362   selection: TMSelection;
018363   selType: TMSelTypes;
```

```
018364   mluField: LONGINT;
018365   END;
018366
018367
018368  { application routines type definitions }
018369  TerminalSendProcPtr = ProcPtr;
018370  TerminalBreakProcPtr = ProcPtr;
018371  TerminalCacheProcPtr = ProcPtr;
018372  TerminalSearchCallBackProcPtr = ProcPtr;
018373  TerminalClikLoopProcPtr = ProcPtr;
018374  TerminalEnvironsProcPtr = ProcPtr;
018375  TerminalChooseIdleProcPtr = ProcPtr;
018376
018377  FUNCTION InitTM: TMErr;
018378  FUNCTION TMGetVersion(hTerm: TermHandle): Handle;
018379  FUNCTION TMGetTMVersion: INTEGER;
018380
018381  FUNCTION TMNew(termRect: Rect;viewRect: Rect;flags: TMFlags;procID: INTEGER;
018382   owner: WindowPtr;sendProc: TerminalSendProcPtr;cacheProc: TerminalCacheProcPtr;
018383   breakProc: TerminalBreakProcPtr;clikLoop: TerminalClikLoopProcPtr;environsProc: TerminalEnvironsProcPtr;
018384   refCon: LONGINT;userData: LONGINT): TermHandle;
018385
018386  PROCEDURE TMDispose(hTerm: TermHandle);
018387
018388  PROCEDURE TMKey(hTerm: TermHandle;theEvent: EventRecord);
018389  PROCEDURE TMUpdate(hTerm: TermHandle;visRgn: RgnHandle);
018390  PROCEDURE TMPaint(hTerm: TermHandle;theTermData: TermDataBlock;theRect: Rect);
018391  PROCEDURE TMActivate(hTerm: TermHandle;activate: BOOLEAN);
018392  PROCEDURE TMResume(hTerm: TermHandle;resume: BOOLEAN);
018393  PROCEDURE TMClick(hTerm: TermHandle;theEvent: EventRecord);
018394  PROCEDURE TMIdle(hTerm: TermHandle);
018395
018396  FUNCTION TMStream(hTerm: TermHandle;theBuffer: Ptr;theLength: LONGINT;flags: CMFlags): LONGINT;
018397  FUNCTION TMMenu(hTerm: TermHandle;menuID: INTEGER;item: INTEGER): BOOLEAN;
018398
018399  PROCEDURE TMReset(hTerm: TermHandle);
018400  PROCEDURE TMClear(hTerm: TermHandle);
018401
018402  PROCEDURE TMResize(hTerm: TermHandle;newViewRect: Rect);
018403
018404  FUNCTION TMGetSelect(hTerm: TermHandle;theData: Handle;VAR theType: ResType): LONGINT;
018405  PROCEDURE TMGetLine(hTerm: TermHandle;lineNo: INTEGER;VAR theTermData: TermDataBlock);
018406  PROCEDURE TMSetSelection(hTerm: TermHandle;theSelection: TMSelection;selType: TMSelTypes);
018407
018408  PROCEDURE TMScroll(hTerm: TermHandle;dh: INTEGER;dv: INTEGER);
018409
018410  FUNCTION TMValidate(hTerm: TermHandle): BOOLEAN;
018411  PROCEDURE TMDefault(VAR theConfig: Ptr;procID: INTEGER;allocate: BOOLEAN);
018412
018413  FUNCTION TMSetupPreflight(procID: INTEGER;VAR magicCookie: LONGINT): Handle;
018414  PROCEDURE TMSetupSetup(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
018415   VAR magicCookie: LONGINT);
018416  FUNCTION TMSetupFilter(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
018417   VAR theEvent: EventRecord;VAR theItem: INTEGER;VAR magicCookie: LONGINT): BOOLEAN;
018418  PROCEDURE TMSetupItem(procID: INTEGER;theConfig: Ptr;count: INTEGER;theDialog: DialogPtr;
018419   VAR theItem: INTEGER;VAR magicCookie: LONGINT);
```

```
018420  PROCEDURE TMSetupXCleanup(procID: INTEGER;theConfig: Ptr;count: INTEGER;
018421   theDialog: DialogPtr;OKed: BOOLEAN;VAR magicCookie: LONGINT);
018422  PROCEDURE TMSetupPostflight(procID: INTEGER);
018423
018424  FUNCTION TMGetConfig(hTerm: TermHandle): Ptr;
018425  FUNCTION TMSetConfig(hTerm: TermHandle;thePtr: Ptr): INTEGER;
018426
018427  FUNCTION TMIntlToEnglish(hTerm: TermHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
018428   language: INTEGER): OSErr;
018429  FUNCTION TMEnglishToIntl(hTerm: TermHandle;inputPtr: Ptr;VAR outputPtr: Ptr;
018430   language: INTEGER): OSErr;
018431
018432  PROCEDURE TMGetToolName(id: INTEGER;VAR name: Str255);
018433  FUNCTION TMGetProcID(name: Str255): INTEGER;
018434
018435  PROCEDURE TMSetRefCon(hTerm: TermHandle;refCon: LONGINT);
018436  FUNCTION TMGetRefCon(hTerm: TermHandle): LONGINT;
018437
018438  PROCEDURE TMSetUserData(hTerm: TermHandle;userData: LONGINT);
018439  FUNCTION TMGetUserData(hTerm: TermHandle): LONGINT;
018440
018441  FUNCTION TMAddSearch(hTerm: TermHandle;theString: Str255;where: Rect;searchType: TMSearchTypes;
018442   callBack: TerminalSearchCallBackProcPtr): INTEGER;
018443  PROCEDURE TMRemoveSearch(hTerm: TermHandle;refnum: INTEGER);
018444  PROCEDURE TMClearSearch(hTerm: TermHandle);
018445
018446  FUNCTION TMGetCursor(hTerm: TermHandle;cursType: TMCursorTypes): Point;
018447
018448  FUNCTION TMGetTermEnvirons(hTerm: TermHandle;VAR theEnvirons: TermEnvironRec): TMErr;
018449
018450  FUNCTION TMChoose(VAR hTerm: TermHandle;where: Point;idleProc: TerminalChooseIdleProcPtr): INTEGER;
018451
018452  PROCEDURE TMEvent(hTerm: TermHandle;theEvent: EventRecord);
018453
018454  FUNCTION TMDoTermKey(hTerm: TermHandle;theKey: Str255): BOOLEAN;
018455  FUNCTION TMCountTermKeys(hTerm: TermHandle): INTEGER;
018456  PROCEDURE TMGetIndTermKey(hTerm: TermHandle;id: INTEGER;VAR theKey: Str255);
018457
018458  PROCEDURE TMGetErrorString(hTerm: TermHandle;id: INTEGER;VAR errMsg: Str255);
018459
018460
018461  {$ENDC} { UsingTerminals }
018462
018463  {$IFC NOT UsingIncludes}
018464   END.
018465  {$ENDC}
018466
018467
018468  ### END OF FILE Terminals.p
018469
```

```
018470
018471  ############################################################################
018472  ### FILE: TerminalTools.p
018473  ############################################################################
018474
018475
018476  {
018477  Created: Thursday, September 12, 1991 at 11:07 AM
018478   TerminalTools.p
018479   Pascal Interface to the Macintosh Libraries
018480
018481    Copyright Apple Computer, Inc. 1988-1991
018482    All rights reserved
018483  }
018484
018485
018486  {$IFC UNDEFINED UsingIncludes}
018487  {$SETC UsingIncludes := 0}
018488  {$ENDC}
018489
018490  {$IFC NOT UsingIncludes}
018491   UNIT TerminalTools;
018492   INTERFACE
018493  {$ENDC}
018494
018495  {$IFC UNDEFINED UsingTerminalTools}
018496  {$SETC UsingTerminalTools := 1}
018497
018498  {$I+}
018499  {$SETC TerminalToolsIncludes := UsingIncludes}
018500  {$SETC UsingIncludes := 1}
018501  {$IFC UNDEFINED UsingDialogs}
018502  {$I $$Shell(PInterfaces)Dialogs.p}
018503  {$ENDC}
018504  {$IFC UNDEFINED UsingTerminals}
018505  {$I $$Shell(PInterfaces)Terminals.p}
018506  {$ENDC}
018507  {$SETC UsingIncludes := TerminalToolsIncludes}
018508
018509  CONST
018510  tdefType = 'tdef';
018511  tvalType = 'tval';
018512  tsetType = 'tset';
018513  tlocType = 'tloc';
018514  tscrType = 'tscr';
018515  tbndType = 'tbnd';
018516  tverType = 'vers';
018517
018518  { messages }
018519  tmInitMsg = 0;
018520  tmDisposeMsg = 1;
018521  tmSuspendMsg = 2;
018522  tmResumeMsg = 3;
018523  tmMenuMsg = 4;
018524  tmEventMsg = 5;
018525  tmActivateMsg = 6;
```

```
018526  tmDeactivateMsg = 7;
018527  tmGetErrorStringMsg = 8;
018528
018529  tmIdleMsg = 50;
018530  tmResetMsg = 51;
018531
018532  tmKeyMsg = 100;
018533  tmStreamMsg = 101;
018534  tmResizeMsg = 102;
018535  tmUpdateMsg = 103;
018536  tmClickMsg = 104;
018537  tmGetSelectionMsg = 105;
018538  tmSetSelectionMsg = 106;
018539  tmScrollMsg = 107;
018540  tmClearMsg = 108;
018541  tmGetLineMsg = 109;
018542  tmPaintMsg = 110;
018543  tmCursorMsg = 111;
018544  tmGetEnvironsMsg = 112;
018545  tmDoTermKeyMsg = 113;
018546  tmCountTermKeysMsg = 114;
018547  tmGetIndTermKeyMsg = 115;
018548
018549  { messages for validate DefProc    }
018550  tmValidateMsg = 0;
018551  tmDefaultMsg = 1;
018552
018553  { messages for Setup DefProc    }
018554  tmSpreflightMsg = 0;
018555  tmSsetupMsg = 1;
018556  tmSitemMsg = 2;
018557  tmSfilterMsg = 3;
018558  tmScleanupMsg = 4;
018559
018560  { messages for scripting defProc    }
018561  tmMgetMsg = 0;
018562  tmMsetMsg = 1;
018563
018564  { messages for localization defProc  }
018565  tmL2English = 0;
018566  tmL2Intl = 1;
018567
018568  TYPE
018569  TMSearchBlockPtr = ^TMSearchBlock;
018570  TMSearchBlock = RECORD
018571   theString: StringHandle;
018572   where: Rect;
018573   searchType: TMSearchTypes;
018574   callBack: ProcPtr;
018575   refnum: INTEGER;
018576   next: TMSearchBlockPtr;
018577   END;
018578
018579  TMSetupPtr = ^TMSetupStruct;
018580  TMSetupStruct = RECORD
018581   theDialog: DialogPtr;
```

```
018582   count: INTEGER;
018583   theConfig: Ptr;
018584   procID: INTEGER;   { procID of the tool }
018585   END;
018586
018587
018588
018589  {$ENDC} { UsingTerminalTools }
018590
018591  {$IFC NOT UsingIncludes}
018592   END.
018593  {$ENDC}
018594
018595
018596  ### END OF FILE TerminalTools.p
018597
```

```
018598
018599  ############################################################################
018600  ### FILE: TextEdit.p
018601  ############################################################################
018602
018603
018604  {
018605  Created: Thursday, September 12, 1991 at 12:34 PM
018606   TextEdit.p
018607   Pascal Interface to the Macintosh Libraries
018608
018609    Copyright Apple Computer, Inc. 1985-1991
018610    All rights reserved
018611  }
018612
018613
018614  {$IFC UNDEFINED UsingIncludes}
018615  {$SETC UsingIncludes := 0}
018616  {$ENDC}
018617
018618  {$IFC NOT UsingIncludes}
018619   UNIT TextEdit;
018620   INTERFACE
018621  {$ENDC}
018622
018623  {$IFC UNDEFINED UsingTextEdit}
018624  {$SETC UsingTextEdit := 1}
018625
018626  {$I+}
018627  {$SETC TextEditIncludes := UsingIncludes}
018628  {$SETC UsingIncludes := 1}
018629  {$IFC UNDEFINED UsingQuickdraw}
018630  {$I $$Shell(PInterfaces)Quickdraw.p}
018631  {$ENDC}
018632  {$SETC UsingIncludes := TextEditIncludes}
018633
018634  CONST
018635
018636  { Justification styles }
018637  teJustLeft = 0;
018638  teJustCenter = 1;
018639  teJustRight = -1;
018640  teForceLeft = -2;
018641
018642  { new names for the Justification styles }
018643  teFlushDefault = 0;                    {flush according to the line direction }
018644  teCenter = 1;                          {center justify }
018645  teFlushRight = -1;                     {flush right for all scripts }
018646  teFlushLeft = -2;                      {flush left for all scripts }
018647
018648  { Set/Replace style modes }
018649  fontBit = 0;                           {set font}
018650  faceBit = 1;                           {set face}
018651  sizeBit = 2;                           {set size}
018652  clrBit = 3;                            {set color}
018653  addSizeBit = 4;                        {add size mode}
```

```
018654  toglBit = 5;                            {set faces in toggle mode}
018655
018656  { TESetStyle/TEContinuousStyle modes }
018657  doFont = 1;                             { set font (family) number}
018658  doFace = 2;                             {set character style}
018659  doSize = 4;                             {set type size}
018660  doColor = 8;                            {set color}
018661  doAll = 15;                             {set all attributes}
018662  addSize = 16;                           {adjust type size}
018663  doToggle = 32;                          {toggle mode for TESetStyle & TEContinuousStyle}
018664
018665  { offsets into TEDispatchRec }
018666  EOLHook = 0;                            {[ProcPtr] TEEOLHook}
018667  DRAWHook = 4;                           {[ProcPtr] TEWidthHook}
018668  WIDTHHook = 8;                          {[ProcPtr] TEDrawHook}
018669  HITTESTHook = 12;                       {[ProcPtr] TEHitTestHook}
018670  nWIDTHHook = 24;                        {[ProcPtr] nTEWidthHook}
018671  TextWidthHook = 28;                     {[ProcPtr] TETextWidthHook}
018672
018673  { selectors for TECustomHook }
018674  intEOLHook = 0;                         {TEIntHook value}
018675  intDrawHook = 1;                        {TEIntHook value}
018676  intWidthHook = 2;                       {TEIntHook value}
018677  intHitTestHook = 3;                     {TEIntHook value}
018678  intNWidthHook = 6;                      {TEIntHook value for new version of WidthHook}
018679  intTextWidthHook = 7;                   {TEIntHook value for new TextWidthHook}
018680
018681  { feature or bit definitions for TEFeatureFlag }
018682  teFAutoScr = 0;                         {00000001b}
018683  teFTextBuffering = 1;                   {00000010b}
018684  teFOutlineHilite = 2;                   {00000100b}
018685  teFInlineInput = 3;                     {00001000b}
018686  teFUseTextServices = 4;                 {00010000b}
018687
018688  { action for the new "bit (un)set" interface, TEFeatureFlag }
018689  TEBitClear = 0;
018690  TEBitSet = 1;                           {set the selector bit}
018691  TEBitTest = -1;                         {no change; just return the current setting}
018692
018693  {constants for identifying the routine that called FindWord }
018694  teWordSelect = 4;                       {clickExpand to select word}
018695  teWordDrag = 8;                         {clickExpand to drag new word}
018696  teFromFind = 12;                        {FindLine called it ($0C)}
018697  teFromRecal = 16;                       {RecalLines called it ($10)}
018698
018699  TYPE
018700  TEPtr = ^TERec;
018701  TEHandle = ^TEPtr;
018702  TERec = RECORD
018703   destRect: Rect;
018704   viewRect: Rect;
018705   selRect: Rect;
018706   lineHeight: INTEGER;
018707   fontAscent: INTEGER;
018708   selPoint: Point;
018709   selStart: INTEGER;
```

```
018710   selEnd: INTEGER;
018711   active: INTEGER;
018712   wordBreak: ProcPtr;
018713   clikLoop: ProcPtr;
018714   clickTime: LONGINT;
018715   clickLoc: INTEGER;
018716   caretTime: LONGINT;
018717   caretState: INTEGER;
018718   just: INTEGER;
018719   teLength: INTEGER;
018720   hText: Handle;
018721   recalBack: INTEGER;
018722   recalLines: INTEGER;
018723   clikStuff: INTEGER;
018724   crOnly: INTEGER;
018725   txFont: INTEGER;
018726   txFace: Style;                        {txFace is unpacked byte}
018727   txMode: INTEGER;
018728   txSize: INTEGER;
018729   inPort: GrafPtr;
018730   highHook: ProcPtr;
018731   caretHook: ProcPtr;
018732   nLines: INTEGER;
018733   lineStarts: ARRAY [0..16000] OF INTEGER;
018734   END;
018735
018736  CharsPtr = ^Chars;
018737  CharsHandle = ^CharsPtr;
018738
018739  Chars = PACKED ARRAY [0..32000] OF CHAR;
018740
018741  StyleRun = RECORD
018742   startChar: INTEGER;                  {starting character position}
018743   styleIndex: INTEGER;                 {index in style table}
018744   END;
018745
018746  STElement = RECORD
018747   stCount: INTEGER;                    {number of runs in this style}
018748   stHeight: INTEGER;                   {line height}
018749   stAscent: INTEGER;                   {font ascent}
018750   stFont: INTEGER;                     {font (family) number}
018751   stFace: Style;                       {character Style}
018752   stSize: INTEGER;                     {size in points}
018753   stColor: RGBColor;                   {absolute (RGB) color}
018754   END;
018755
018756  STPtr = ^TEStyleTable;
018757  STHandle = ^STPtr;
018758
018759  TEStyleTable = ARRAY [0..1776] OF STElement;
018760
018761  LHElement = RECORD
018762   lhHeight: INTEGER;                   {maximum height in line}
018763   lhAscent: INTEGER;                   {maximum ascent in line}
018764   END;
018765
```

```
018766  LHPtr = ^LHTable;
018767  LHHandle = ^LHPtr;
018768
018769  LHTable = ARRAY [0..8000] OF LHElement;
018770
018771  ScrpSTElement = RECORD
018772   scrpStartChar: LONGINT;                {starting character position}
018773   scrpHeight: INTEGER;                   {starting character position}
018774   scrpAscent: INTEGER;
018775   scrpFont: INTEGER;
018776   scrpFace: Style;                       {unpacked byte}
018777   scrpSize: INTEGER;
018778   scrpColor: RGBColor;
018779   END;
018780
018781  ScrpSTTable = ARRAY[0..1600] OF ScrpSTElement;
018782
018783  StScrpPtr = ^StScrpRec;
018784  StScrpHandle = ^StScrpPtr;
018785  StScrpRec = RECORD
018786   scrpNStyles: INTEGER;                  {number of styles in scrap}
018787   scrpStyleTab: ScrpSTTable;             {table of styles for scrap}
018788   END;
018789
018790  NullStPtr = ^NullStRec;
018791  NullStHandle = ^NullStPtr;
018792  NullStRec = RECORD
018793   teReserved: LONGINT;                   {reserved for future expansion}
018794   nullScrap: StScrpHandle;               {handle to scrap style table}
018795   END;
018796
018797  TEStylePtr = ^TEStyleRec;
018798  TEStyleHandle = ^TEStylePtr;
018799  TEStyleRec = RECORD
018800   nRuns: INTEGER;                        {number of style runs}
018801   nStyles: INTEGER;                      {size of style table}
018802   styleTab: STHandle;                    {handle to style table}
018803   lhTab: LHHandle;                       {handle to line-height table}
018804   teRefCon: LONGINT;                     {reserved for application use}
018805   nullStyle: NullStHandle;               {Handle to style set at null selection}
018806   runs: ARRAY [0..8000] OF StyleRun;     {ARRAY [0..8000] OF StyleRun}
018807   END;
018808
018809  TextStylePtr = ^TextStyle;
018810  TextStyleHandle = ^TextStylePtr;
018811  TextStyle = RECORD
018812   tsFont: INTEGER;                       {font (family) number}
018813   tsFace: Style;                         {character Style}
018814   tsSize: INTEGER;                       {size in point}
018815   tsColor: RGBColor;                     {absolute (RGB) color}
018816   END;
018817
018818
018819  TEIntHook = INTEGER;
018820
018821  PROCEDURE TEInit;
```

```
018822   INLINE $A9CC;
018823  FUNCTION TENew(destRect: Rect;viewRect: Rect): TEHandle;
018824   INLINE $A9D2;
018825  PROCEDURE TEDispose(hTE: TEHandle);
018826   INLINE $A9CD;
018827  PROCEDURE TESetText(text: Ptr;length: LONGINT;hTE: TEHandle);
018828   INLINE $A9CF;
018829  FUNCTION TEGetText(hTE: TEHandle): CharsHandle;
018830   INLINE $A9CB;
018831  PROCEDURE TEIdle(hTE: TEHandle);
018832   INLINE $A9DA;
018833  PROCEDURE TESetSelect(selStart: LONGINT;selEnd: LONGINT;hTE: TEHandle);
018834   INLINE $A9D1;
018835  PROCEDURE TEActivate(hTE: TEHandle);
018836   INLINE $A9D8;
018837  PROCEDURE TEDeactivate(hTE: TEHandle);
018838   INLINE $A9D9;
018839  PROCEDURE TEKey(key: CHAR;hTE: TEHandle);
018840   INLINE $A9DC;
018841  PROCEDURE TECut(hTE: TEHandle);
018842   INLINE $A9D6;
018843  PROCEDURE TECopy(hTE: TEHandle);
018844   INLINE $A9D5;
018845  PROCEDURE TEPaste(hTE: TEHandle);
018846   INLINE $A9DB;
018847  PROCEDURE TEDelete(hTE: TEHandle);
018848   INLINE $A9D7;
018849  PROCEDURE TEInsert(text: Ptr;length: LONGINT;hTE: TEHandle);
018850   INLINE $A9DE;
018851  PROCEDURE TESetJust(just: INTEGER;hTE: TEHandle);
018852   INLINE $A9DF;
018853  PROCEDURE TEUpdate(rUpdate: Rect;hTE: TEHandle);
018854   INLINE $A9D3;
018855  PROCEDURE TextBox(text: Ptr;length: LONGINT;box: Rect;just: INTEGER);
018856   INLINE $A9CE;
018857  PROCEDURE TEScroll(dh: INTEGER;dv: INTEGER;hTE: TEHandle);
018858   INLINE $A9DD;
018859  PROCEDURE TESelView(hTE: TEHandle);
018860   INLINE $A811;
018861  PROCEDURE TEPinScroll(dh: INTEGER;dv: INTEGER;hTE: TEHandle);
018862   INLINE $A812;
018863  PROCEDURE TEAutoView(fAuto: BOOLEAN;hTE: TEHandle);
018864   INLINE $A813;
018865  FUNCTION TEScrapHandle: Handle;
018866   INLINE $2EB8,$0AB4;
018867  PROCEDURE TECalText(hTE: TEHandle);
018868   INLINE $A9D0;
018869  FUNCTION TEGetOffset(pt: Point;hTE: TEHandle): INTEGER;
018870   INLINE $A83C;
018871  FUNCTION TEGetPoint(offset: INTEGER;hTE: TEHandle): Point;
018872   INLINE $3F3C,$0008,$A83D;
018873  PROCEDURE TEClick(pt: Point;fExtend: BOOLEAN;h: TEHandle);
018874   INLINE $A9D4;
018875  FUNCTION TEStylNew(destRect: Rect;viewRect: Rect): TEHandle;
018876   INLINE $A83E;
018877  FUNCTION TEStyleNew(destRect: Rect;viewRect: Rect): TEHandle;
```

```
018878   INLINE $A83E;
018879  PROCEDURE SetStylHandle(theHandle: TEStyleHandle;hTE: TEHandle);
018880   INLINE $3F3C,$0005,$A83D;
018881  PROCEDURE SetStyleHandle(theHandle: TEStyleHandle;hTE: TEHandle);
018882   INLINE $3F3C,$0005,$A83D;
018883  FUNCTION GetStylHandle(hTE: TEHandle): TEStyleHandle;
018884   INLINE $3F3C,$0004,$A83D;
018885  FUNCTION GetStyleHandle(hTE: TEHandle): TEStyleHandle;
018886   INLINE $3F3C,$0004,$A83D;
018887  PROCEDURE TEGetStyle(offset: INTEGER;VAR theStyle: TextStyle;VAR lineHeight: INTEGER;
018888   VAR fontAscent: INTEGER;hTE: TEHandle);
018889   INLINE $3F3C,$0003,$A83D;
018890  PROCEDURE TEStylPaste(hTE: TEHandle);
018891   INLINE $3F3C,$0000,$A83D;
018892  PROCEDURE TEStylePaste(hTE: TEHandle);
018893   INLINE $3F3C,$0000,$A83D;
018894  PROCEDURE TESetStyle(mode: INTEGER;newStyle: TextStyle;redraw: BOOLEAN;
018895   hTE: TEHandle);
018896   INLINE $3F3C,$0001,$A83D;
018897  PROCEDURE TEReplaceStyle(mode: INTEGER;oldStyle: TextStyle;newStyle: TextStyle;
018898   redraw: BOOLEAN;hTE: TEHandle);
018899   INLINE $3F3C,$0002,$A83D;
018900  FUNCTION GetStylScrap(hTE: TEHandle): StScrpHandle;
018901   INLINE $3F3C,$0006,$A83D;
018902  FUNCTION GetStyleScrap(hTE: TEHandle): StScrpHandle;
018903   INLINE $3F3C,$0006,$A83D;
018904  PROCEDURE TEStylInsert(text: Ptr;length: LONGINT;hST: StScrpHandle;hTE: TEHandle);
018905   INLINE $3F3C,$0007,$A83D;
018906  PROCEDURE TEStyleInsert(text: Ptr;length: LONGINT;hST: StScrpHandle;hTE: TEHandle);
018907   INLINE $3F3C,$0007,$A83D;
018908  FUNCTION TEGetHeight(endLine: LONGINT;startLine: LONGINT;hTE: TEHandle): LONGINT;
018909   INLINE $3F3C,$0009,$A83D;
018910  FUNCTION TEContinuousStyle(VAR mode: INTEGER;VAR aStyle: TextStyle;hTE: TEHandle): BOOLEAN;
018911   INLINE $3F3C,$000A,$A83D;
018912  PROCEDURE SetStylScrap(rangeStart: LONGINT;rangeEnd: LONGINT;newStyles: StScrpHandle;
018913   redraw: BOOLEAN;hTE: TEHandle);
018914   INLINE $3F3C,$000B,$A83D;
018915  PROCEDURE SetStyleScrap(rangeStart: LONGINT;rangeEnd: LONGINT;newStyles: StScrpHandle;
018916   redraw: BOOLEAN;hTE: TEHandle);
018917   INLINE $3F3C,$000B,$A83D;
018918  PROCEDURE TECustomHook(which: TEIntHook;VAR addr: ProcPtr;hTE: TEHandle);
018919   INLINE $3F3C,$000C,$A83D;
018920  FUNCTION TENumStyles(rangeStart: LONGINT;rangeEnd: LONGINT;hTE: TEHandle): LONGINT;
018921   INLINE $3F3C,$000D,$A83D;
018922  FUNCTION TEFeatureFlag(feature: INTEGER;action: INTEGER;hTE: TEHandle): INTEGER;
018923   INLINE $3F3C,$000E,$A83D;
018924  FUNCTION TEGetScrapLen: LONGINT;
018925  PROCEDURE TESetScrapLen(length: LONGINT);
018926  FUNCTION TEFromScrap: OSErr;
018927  FUNCTION TEToScrap: OSErr;
018928  PROCEDURE SetClikLoop(clikProc: ProcPtr;hTE: TEHandle);
018929  PROCEDURE SetWordBreak(wBrkProc: ProcPtr;hTE: TEHandle);
018930
018931
018932  {$ENDC} { UsingTextEdit }
018933
```

```
018934  {$IFC NOT UsingIncludes}
018935   END.
018936  {$ENDC}
018937
018938
018939  ### END OF FILE TextEdit.p
018940
```

```
018941
018942  ##########################################################################
018943  ### FILE: Timer.p
018944  ##########################################################################
018945
018946  {
018947  Created: Sunday, January 6, 1991 at 11:25 PM
018948      Timer.p
018949      Pascal Interface to the Macintosh Libraries
018950
018951          Copyright Apple Computer, Inc.    1985-1990
018952          All rights reserved
018953  }
018954
018955
018956  {$IFC UNDEFINED UsingIncludes}
018957  {$SETC UsingIncludes := 0}
018958  {$ENDC}
018959
018960  {$IFC NOT UsingIncludes}
018961      UNIT Timer;
018962      INTERFACE
018963  {$ENDC}
018964
018965  {$IFC UNDEFINED UsingTimer}
018966  {$SETC UsingTimer := 1}
018967
018968  {$I+}
018969  {$SETC TimerIncludes := UsingIncludes}
018970  {$SETC UsingIncludes := 1}
018971  {$IFC UNDEFINED UsingTypes}
018972  {$I $$Shell(PInterfaces)Types.p}
018973  {$ENDC}
018974  {$IFC UNDEFINED UsingOSUtils}
018975  {$I $$Shell(PInterfaces)OSUtils.p}
018976  {$ENDC}
018977  {$SETC UsingIncludes := TimerIncludes}
018978
018979  TYPE
018980  TMTaskPtr = ^TMTask;
018981  TMTask = RECORD
018982      qLink: QElemPtr;
018983      qType: INTEGER;
018984      tmAddr: ProcPtr;
018985      tmCount: LONGINT;
018986      tmWakeUp: LONGINT;
018987      tmReserved: LONGINT;
018988      END;
018989
018990
018991  PROCEDURE InsTime(tmTaskPtr: QElemPtr);
018992      INLINE $205F,$A058;
018993  PROCEDURE InsXTime(tmTaskPtr: QElemPtr);
018994      INLINE $205F,$A458;
018995  PROCEDURE PrimeTime(tmTaskPtr: QElemPtr;count: LONGINT);
018996      INLINE $201F,$205F,$A05A;
```

```
018997  PROCEDURE RmvTime(tmTaskPtr: QElemPtr);
018998      INLINE $205F,$A059;
018999
019000
019001  {$ENDC}    { UsingTimer }
019002
019003  {$IFC NOT UsingIncludes}
019004      END.
019005  {$ENDC}
019006
019007
019008  ### END OF FILE Timer.p
019009
```

```
019010
019011  ############################################################################
019012  ### FILE: ToolIntf.p
019013  ############################################################################
019014
019015  {
019016      File: ToolIntf.p
019017
019018      As of MPW 3.0, interface files were reorganized to more closely
019019      match "Inside Macintosh" reference books and be more consistant
019020      from language to language.
019021
019022      Interfaces for the Macintosh toolbox calls are now found in the
019023      files included below. This file is provided for compatibility
019024      with old sources.
019025
019026      Pascal Interface to the Macintosh Libraries
019027      Copyright Apple Computer, Inc. 1988
019028      All Rights Reserved
019029  }
019030
019031  {$IFC UNDEFINED UsingIncludes}
019032  {$SETC UsingIncludes := 0}
019033  {$ENDC}
019034
019035  {$IFC NOT UsingIncludes}
019036      UNIT ToolIntf;
019037      INTERFACE
019038  {$ENDC}
019039
019040  {$IFC UNDEFINED UsingToolIntf}
019041  {$SETC UsingToolIntf := 1}
019042
019043  {$I+}
019044  {$SETC ToolIntfIncludes := UsingIncludes}
019045  {$SETC UsingIncludes := 1}
019046  {$IFC UNDEFINED UsingEvents}
019047  {$I $$Shell(PInterfaces)Events.p}
019048  {$ENDC}
019049  {$IFC UNDEFINED UsingControls}
019050  {$I $$Shell(PInterfaces)Controls.p}
019051  {$ENDC}
019052  {$IFC UNDEFINED UsingDesk}
019053  {$I $$Shell(PInterfaces)Desk.p}
019054  {$ENDC}
019055  {$IFC UNDEFINED UsingWindows}
019056  {$I $$Shell(PInterfaces)Windows.p}
019057  {$ENDC}
019058  {$IFC UNDEFINED UsingTextEdit}
019059  {$I $$Shell(PInterfaces)TextEdit.p}
019060  {$ENDC}
019061  {$IFC UNDEFINED UsingDialogs}
019062  {$I $$Shell(PInterfaces)Dialogs.p}
019063  {$ENDC}
019064  {$IFC UNDEFINED UsingFonts}
019065  {$I $$Shell(PInterfaces)Fonts.p}
```

```
019066  {$ENDC}
019067  {$IFC UNDEFINED UsingLists}
019068  {$I $$Shell(PInterfaces)Lists.p}
019069  {$ENDC}
019070  {$IFC UNDEFINED UsingMenus}
019071  {$I $$Shell(PInterfaces)Menus.p}
019072  {$ENDC}
019073  {$IFC UNDEFINED UsingResources}
019074  {$I $$Shell(PInterfaces)Resources.p}
019075  {$ENDC}
019076  {$IFC UNDEFINED UsingScrap}
019077  {$I $$Shell(PInterfaces)Scrap.p}
019078  {$ENDC}
019079  {$IFC UNDEFINED UsingToolUtils}
019080  {$I $$Shell(PInterfaces)ToolUtils.p}
019081  {$ENDC}
019082  {$SETC UsingIncludes := ToolIntfIncludes}
019083
019084  {$ENDC}    { UsingToolIntf }
019085
019086  {$IFC NOT UsingIncludes}
019087      END.
019088  {$ENDC}
019089
019090
019091  ### END OF FILE ToolIntf.p
019092
```

```
019093
019094  ##########################################################################
019095  ### FILE: ToolUtils.p
019096  ##########################################################################
019097
019098  {
019099  Created: Sunday, January 6, 1991 at 11:25 PM
019100      ToolUtils.p
019101      Pascal Interface to the Macintosh Libraries
019102
019103          Copyright Apple Computer, Inc.    1985-1990
019104          All rights reserved
019105  }
019106
019107
019108  {$IFC UNDEFINED UsingIncludes}
019109  {$SETC UsingIncludes := 0}
019110  {$ENDC}
019111
019112  {$IFC NOT UsingIncludes}
019113      UNIT ToolUtils;
019114      INTERFACE
019115  {$ENDC}
019116
019117  {$IFC UNDEFINED UsingToolUtils}
019118  {$SETC UsingToolUtils := 1}
019119
019120  {$I+}
019121  {$SETC ToolUtilsIncludes := UsingIncludes}
019122  {$SETC UsingIncludes := 1}
019123  {$IFC UNDEFINED UsingQuickdraw}
019124  {$I $$Shell(PInterfaces)Quickdraw.p}
019125  {$ENDC}
019126  {$SETC UsingIncludes := ToolUtilsIncludes}
019127
019128  CONST
019129  sysPatListID = 0;
019130  iBeamCursor = 1;
019131  crossCursor = 2;
019132  plusCursor = 3;
019133  watchCursor = 4;
019134
019135  TYPE
019136  Int64Bit = RECORD
019137      hiLong: LONGINT;
019138      loLong: LONGINT;
019139      END;
019140
019141
019142  FUNCTION FixRatio(numer: INTEGER;denom: INTEGER): Fixed;
019143      INLINE $A869;
019144  FUNCTION FixMul(a: Fixed;b: Fixed): Fixed;
019145      INLINE $A868;
019146  FUNCTION FixRound(x: Fixed): INTEGER;
019147      INLINE $A86C;
019148  FUNCTION GetString(stringID: INTEGER): StringHandle;
```

```
019149      INLINE $A9BA;
019150  FUNCTION Munger(h: Handle;offset: LONGINT;ptr1: Ptr;len1: LONGINT;ptr2: Ptr;
019151      len2: LONGINT): LONGINT;
019152      INLINE $A9E0;
019153  PROCEDURE PackBits(VAR srcPtr: Ptr;VAR dstPtr: Ptr;srcBytes: INTEGER);
019154      INLINE $A8CF;
019155  PROCEDURE UnpackBits(VAR srcPtr: Ptr;VAR dstPtr: Ptr;dstBytes: INTEGER);
019156      INLINE $A8D0;
019157  FUNCTION BitTst(bytePtr: Ptr;bitNum: LONGINT): BOOLEAN;
019158      INLINE $A85D;
019159  PROCEDURE BitSet(bytePtr: Ptr;bitNum: LONGINT);
019160      INLINE $A85E;
019161  PROCEDURE BitClr(bytePtr: Ptr;bitNum: LONGINT);
019162      INLINE $A85F;
019163  FUNCTION BitAnd(value1: LONGINT;value2: LONGINT): LONGINT;
019164      INLINE $A858;
019165  FUNCTION BitOr(value1: LONGINT;value2: LONGINT): LONGINT;
019166      INLINE $A85B;
019167  FUNCTION BitXor(value1: LONGINT;value2: LONGINT): LONGINT;
019168      INLINE $A859;
019169  FUNCTION BitNot(value: LONGINT): LONGINT;
019170      INLINE $A85A;
019171  FUNCTION BitShift(value: LONGINT;count: INTEGER): LONGINT;
019172      INLINE $A85C;
019173  FUNCTION HiWord(x: LONGINT): INTEGER;
019174      INLINE $A86A;
019175  FUNCTION LoWord(x: LONGINT): INTEGER;
019176      INLINE $A86B;
019177  PROCEDURE LongMul(a: LONGINT;b: LONGINT;VAR result: Int64Bit);
019178      INLINE $A867;
019179  FUNCTION GetIcon(iconID: INTEGER): Handle;
019180      INLINE $A9BB;
019181  PROCEDURE PlotIcon(theRect: Rect;theIcon: Handle);
019182      INLINE $A94B;
019183  FUNCTION GetPattern(patternID: INTEGER): PatHandle;
019184      INLINE $A9B8;
019185  FUNCTION GetCursor(cursorID: INTEGER): CursHandle;
019186      INLINE $A9B9;
019187  FUNCTION GetPicture(pictureID: INTEGER): PicHandle;
019188      INLINE $A9BC;
019189  FUNCTION SlopeFromAngle(angle: INTEGER): Fixed;
019190      INLINE $A8BC;
019191  FUNCTION AngleFromSlope(slope: Fixed): INTEGER;
019192      INLINE $A8C4;
019193  PROCEDURE SetString(theString: StringHandle;strNew: Str255);
019194      INLINE $A907;
019195  FUNCTION DeltaPoint(ptA: Point;ptB: Point): LONGINT;
019196      INLINE $A94F;
019197  FUNCTION NewString(theString: Str255): StringHandle;
019198      INLINE $A906;
019199  PROCEDURE ShieldCursor(shieldRect: Rect;offsetPt: Point);
019200      INLINE $A855;
019201  PROCEDURE GetIndString(VAR theString: Str255;strListID: INTEGER;index: INTEGER);
019202  PROCEDURE ScreenRes(VAR scrnHRes: INTEGER;VAR scrnVRes: INTEGER);
019203      INLINE $225F,$32B8,$0102,$225F,$32B8,$0104;
019204  PROCEDURE GetIndPattern(VAR thePat: Pattern;patternListID: INTEGER;index: INTEGER);
```

```
019205
019206
019207  {$ENDC}    { UsingToolUtils }
019208
019209  {$IFC NOT UsingIncludes}
019210      END.
019211  {$ENDC}
019212
019213
019214  ### END OF FILE ToolUtils.p
019215
```

```
019216
019217  ##########################################################################
019218  ### FILE: Traps.p
019219  ##########################################################################
019220
019221
019222  {
019223  Created: Saturday, December 7, 1991 at 12:50 PM
019224   Traps.p
019225   Pascal Interface to the Macintosh Libraries
019226
019227    Copyright Apple Computer, Inc. 1986-1991
019228    All rights reserved
019229  }
019230
019231
019232  {$IFC UNDEFINED UsingIncludes}
019233  {$SETC UsingIncludes := 0}
019234  {$ENDC}
019235
019236  {$IFC NOT UsingIncludes}
019237   UNIT Traps;
019238   INTERFACE
019239  {$ENDC}
019240
019241  {$IFC UNDEFINED UsingTraps}
019242  {$SETC UsingTraps := 1}
019243
019244  CONST
019245
019246  {
019247  ; QuickDraw
019248
019249   }
019250  _CopyMask = $A817;
019251  _MeasureText = $A837;
019252  _GetMaskTable = $A836;
019253  _CalcMask = $A838;
019254  _SeedFill = $A839;
019255  _InitCursor = $A850;
019256  _SetCursor = $A851;
019257  _HideCursor = $A852;
019258  _ShowCursor = $A853;
019259  _ShieldCursor = $A855;
019260  _ObscureCursor = $A856;
019261  _BitAnd = $A858;
019262  _BitXOr = $A859;
019263  _BitNot = $A85A;
019264  _BitOr = $A85B;
019265  _BitShift = $A85C;
019266  _BitTst = $A85D;
019267  _BitSet = $A85E;
019268  _BitClr = $A85F;
019269  _Random = $A861;
019270  _ForeColor = $A862;
019271  _BackColor = $A863;
```

```
019272  _ColorBit = $A864;
019273  _GetPixel = $A865;
019274  _StuffHex = $A866;
019275  _LongMul = $A867;
019276  _FixMul = $A868;
019277  _FixRatio = $A869;
019278  _HiWord = $A86A;
019279  _LoWord = $A86B;
019280  _FixRound = $A86C;
019281  _InitPort = $A86D;
019282  _InitGraf = $A86E;
019283  _OpenPort = $A86F;
019284  _LocalToGlobal = $A870;
019285  _GlobalToLocal = $A871;
019286  _GrafDevice = $A872;
019287  _SetPort = $A873;
019288  _GetPort = $A874;
019289  _SetPBits = $A875;
019290  _PortSize = $A876;
019291  _MovePortTo = $A877;
019292  _SetOrigin = $A878;
019293  _SetClip = $A879;
019294  _GetClip = $A87A;
019295  _ClipRect = $A87B;
019296  _BackPat = $A87C;
019297  _ClosePort = $A87D;
019298  _AddPt = $A87E;
019299  _SubPt = $A87F;
019300  _SetPt = $A880;
019301  _EqualPt = $A881;
019302  _StdText = $A882;
019303  _DrawChar = $A883;
019304  _DrawString = $A884;
019305  _DrawText = $A885;
019306  _TextWidth = $A886;
019307  _TextFont = $A887;
019308  _TextFace = $A888;
019309  _TextMode = $A889;
019310  _TextSize = $A88A;
019311  _GetFontInfo = $A88B;
019312  _StringWidth = $A88C;
019313  _CharWidth = $A88D;
019314  _SpaceExtra = $A88E;
019315  _StdLine = $A890;
019316  _LineTo = $A891;
019317  _Line = $A892;
019318  _MoveTo = $A893;
019319  _Move = $A894;
019320  _ShutDown = $A895;
019321  _HidePen = $A896;
019322  _ShowPen = $A897;
019323  _GetPenState = $A898;
019324  _SetPenState = $A899;
019325  _GetPen = $A89A;
019326  _PenSize = $A89B;
019327  _PenMode = $A89C;
```

```
019328  _PenPat = $A89D;
019329  _PenNormal = $A89E;
019330  _Unimplemented = $A89F;
019331  _StdRect = $A8A0;
019332  _FrameRect = $A8A1;
019333  _PaintRect = $A8A2;
019334  _EraseRect = $A8A3;
019335  _InverRect = $A8A4;
019336  _FillRect = $A8A5;
019337  _EqualRect = $A8A6;
019338  _SetRect = $A8A7;
019339  _OffsetRect = $A8A8;
019340  _InsetRect = $A8A9;
019341  _SectRect = $A8AA;
019342  _UnionRect = $A8AB;
019343  _Pt2Rect = $A8AC;
019344  _PtInRect = $A8AD;
019345  _EmptyRect = $A8AE;
019346  _StdRRect = $A8AF;
019347  _FrameRoundRect = $A8B0;
019348  _PaintRoundRect = $A8B1;
019349  _EraseRoundRect = $A8B2;
019350  _InverRoundRect = $A8B3;
019351  _FillRoundRect = $A8B4;
019352  _StdOval = $A8B6;
019353  _FrameOval = $A8B7;
019354  _PaintOval = $A8B8;
019355  _EraseOval = $A8B9;
019356  _InvertOval = $A8BA;
019357  _FillOval = $A8BB;
019358  _SlopeFromAngle = $A8BC;
019359  _StdArc = $A8BD;
019360  _FrameArc = $A8BE;
019361  _PaintArc = $A8BF;
019362  _EraseArc = $A8C0;
019363  _InvertArc = $A8C1;
019364  _FillArc = $A8C2;
019365  _PtToAngle = $A8C3;
019366  _AngleFromSlope = $A8C4;
019367  _StdPoly = $A8C5;
019368  _FramePoly = $A8C6;
019369  _PaintPoly = $A8C7;
019370  _ErasePoly = $A8C8;
019371  _InvertPoly = $A8C9;
019372  _FillPoly = $A8CA;
019373  _OpenPoly = $A8CB;
019374  _ClosePgon = $A8CC;
019375  _ClosePoly = $A8CC;
019376  _KillPoly = $A8CD;
019377  _OffsetPoly = $A8CE;
019378  _PackBits = $A8CF;
019379  _UnpackBits = $A8D0;
019380  _StdRgn = $A8D1;
019381  _FrameRgn = $A8D2;
019382  _PaintRgn = $A8D3;
019383  _EraseRgn = $A8D4;
```

```
019384  _InverRgn = $A8D5;
019385  _FillRgn = $A8D6;
019386  _BitMapRgn = $A8D7;
019387  _BitMapToRegion = $A8D7;
019388  _NewRgn = $A8D8;
019389  _DisposRgn = $A8D9;
019390  _DisposeRgn = $A8D9;
019391  _OpenRgn = $A8DA;
019392  _CloseRgn = $A8DB;
019393  _CopyRgn = $A8DC;
019394  _SetEmptyRgn = $A8DD;
019395  _SetRecRgn = $A8DE;
019396  _RectRgn = $A8DF;
019397  _OfsetRgn = $A8E0;
019398  _OffsetRgn = $A8E0;
019399  _InsetRgn = $A8E1;
019400  _EmptyRgn = $A8E2;
019401  _EqualRgn = $A8E3;
019402  _SectRgn = $A8E4;
019403  _UnionRgn = $A8E5;
019404  _DiffRgn = $A8E6;
019405  _XOrRgn = $A8E7;
019406  _PtInRgn = $A8E8;
019407  _RectInRgn = $A8E9;
019408  _SetStdProcs = $A8EA;
019409  _StdBits = $A8EB;
019410  _CopyBits = $A8EC;
019411  _StdTxMeas = $A8ED;
019412  _StdGetPic = $A8EE;
019413  _ScrollRect = $A8EF;
019414  _StdPutPic = $A8F0;
019415  _StdComment = $A8F1;
019416  _PicComment = $A8F2;
019417  _OpenPicture = $A8F3;
019418  _ClosePicture = $A8F4;
019419  _KillPicture = $A8F5;
019420  _DrawPicture = $A8F6;
019421  _Layout = $A8F7;
019422  _ScalePt = $A8F8;
019423  _MapPt = $A8F9;
019424  _MapRect = $A8FA;
019425  _MapRgn = $A8FB;
019426  _MapPoly = $A8FC;
019427
019428  {
019429  ; Toolbox
019430
019431   }
019432  _Count1Resources = $A80D;
019433  _Get1IxResource = $A80E;
019434  _Get1IxType = $A80F;
019435  _Unique1ID = $A810;
019436  _TESelView = $A811;
019437  _TEPinScroll = $A812;
019438  _TEAutoView = $A813;
019439  _Pack8 = $A816;
```

```
019440  _FixATan2 = $A818;
019441  _XMunger = $A819;
019442  _HOpenResFile = $A81A;
019443  _HCreateResFile = $A81B;
019444  _Count1Types = $A81C;
019445  _Get1Resource = $A81F;
019446  _Get1NamedResource = $A820;
019447  _MaxSizeRsrc = $A821;
019448  _InsMenuItem = $A826;
019449  _HideDItem = $A827;
019450  _ShowDItem = $A828;
019451  _LayerDispatch = $A829;
019452  _Pack9 = $A82B;
019453  _Pack10 = $A82C;
019454  _Pack11 = $A82D;
019455  _Pack12 = $A82E;
019456  _Pack13 = $A82F;
019457  _Pack14 = $A830;
019458  _Pack15 = $A831;
019459  _ScrnBitMap = $A833;
019460  _SetFScaleDisable = $A834;
019461  _FontMetrics = $A835;
019462  _ZoomWindow = $A83A;
019463  _TrackBox = $A83B;
019464  _PrGlue = $A8FD;
019465  _InitFonts = $A8FE;
019466  _GetFName = $A8FF;
019467  _GetFNum = $A900;
019468  _FMSwapFont = $A901;
019469  _RealFont = $A902;
019470  _SetFontLock = $A903;
019471  _DrawGrowIcon = $A904;
019472  _DragGrayRgn = $A905;
019473  _NewString = $A906;
019474  _SetString = $A907;
019475  _ShowHide = $A908;
019476  _CalcVis = $A909;
019477  _CalcVBehind = $A90A;
019478  _ClipAbove = $A90B;
019479  _PaintOne = $A90C;
019480  _PaintBehind = $A90D;
019481  _SaveOld = $A90E;
019482  _DrawNew = $A90F;
019483  _GetWMgrPort = $A910;
019484  _CheckUpDate = $A911;
019485  _InitWindows = $A912;
019486  _NewWindow = $A913;
019487  _DisposWindow = $A914;
019488  _DisposeWindow = $A914;
019489  _ShowWindow = $A915;
019490  _HideWindow = $A916;
019491  _GetWRefCon = $A917;
019492  _SetWRefCon = $A918;
019493  _GetWTitle = $A919;
019494  _SetWTitle = $A91A;
019495  _MoveWindow = $A91B;
```

```
019496  _HiliteWindow = $A91C;
019497  _SizeWindow = $A91D;
019498  _TrackGoAway = $A91E;
019499  _SelectWindow = $A91F;
019500  _BringToFront = $A920;
019501  _SendBehind = $A921;
019502  _BeginUpDate = $A922;
019503  _EndUpDate = $A923;
019504  _FrontWindow = $A924;
019505  _DragWindow = $A925;
019506  _DragTheRgn = $A926;
019507  _InvalRgn = $A927;
019508  _InvalRect = $A928;
019509  _ValidRgn = $A929;
019510  _ValidRect = $A92A;
019511  _GrowWindow = $A92B;
019512  _FindWindow = $A92C;
019513  _CloseWindow = $A92D;
019514  _SetWindowPic = $A92E;
019515  _GetWindowPic = $A92F;
019516  _InitMenus = $A930;
019517  _NewMenu = $A931;
019518  _DisposMenu = $A932;
019519  _DisposeMenu = $A932;
019520  _AppendMenu = $A933;
019521  _ClearMenuBar = $A934;
019522  _InsertMenu = $A935;
019523  _DeleteMenu = $A936;
019524  _DrawMenuBar = $A937;
019525  _InvalMenuBar = $A81D;
019526  _HiliteMenu = $A938;
019527  _EnableItem = $A939;
019528  _DisableItem = $A93A;
019529  _GetMenuBar = $A93B;
019530  _SetMenuBar = $A93C;
019531  _MenuSelect = $A93D;
019532  _MenuKey = $A93E;
019533  _GetItmIcon = $A93F;
019534  _SetItmIcon = $A940;
019535  _GetItmStyle = $A941;
019536  _SetItmStyle = $A942;
019537  _GetItmMark = $A943;
019538  _SetItmMark = $A944;
019539  _CheckItem = $A945;
019540  _GetItem = $A946;
019541  _SetItem = $A947;
019542  _CalcMenuSize = $A948;
019543  _GetMHandle = $A949;
019544  _SetMFlash = $A94A;
019545  _PlotIcon = $A94B;
019546  _FlashMenuBar = $A94C;
019547  _AddResMenu = $A94D;
019548  _PinRect = $A94E;
019549  _DeltaPoint = $A94F;
019550  _CountMItems = $A950;
019551  _InsertResMenu = $A951;
```

```
019552  _DelMenuItem = $A952;
019553  _UpdtControl = $A953;
019554  _NewControl = $A954;
019555  _DisposControl = $A955;
019556  _DisposeControl = $A955;
019557  _KillControls = $A956;
019558  _ShowControl = $A957;
019559  _HideControl = $A958;
019560  _MoveControl = $A959;
019561  _GetCRefCon = $A95A;
019562  _SetCRefCon = $A95B;
019563  _SizeControl = $A95C;
019564  _HiliteControl = $A95D;
019565  _GetCTitle = $A95E;
019566  _SetCTitle = $A95F;
019567  _GetCtlValue = $A960;
019568  _GetMinCtl = $A961;
019569  _GetMaxCtl = $A962;
019570  _SetCtlValue = $A963;
019571  _SetMinCtl = $A964;
019572  _SetMaxCtl = $A965;
019573  _TestControl = $A966;
019574  _DragControl = $A967;
019575  _TrackControl = $A968;
019576  _DrawControls = $A969;
019577  _GetCtlAction = $A96A;
019578  _SetCtlAction = $A96B;
019579  _FindControl = $A96C;
019580  _Draw1Control = $A96D;
019581  _Dequeue = $A96E;
019582  _Enqueue = $A96F;
019583  _WaitNextEvent = $A860;
019584  _GetNextEvent = $A970;
019585  _EventAvail = $A971;
019586  _GetMouse = $A972;
019587  _StillDown = $A973;
019588  _Button = $A974;
019589  _TickCount = $A975;
019590  _GetKeys = $A976;
019591  _WaitMouseUp = $A977;
019592  _UpdtDialog = $A978;
019593  _CouldDialog = $A979;
019594  _FreeDialog = $A97A;
019595  _InitDialogs = $A97B;
019596  _GetNewDialog = $A97C;
019597  _NewDialog = $A97D;
019598  _SelIText = $A97E;
019599  _IsDialogEvent = $A97F;
019600  _DialogSelect = $A980;
019601  _DrawDialog = $A981;
019602  _CloseDialog = $A982;
019603  _DisposDialog = $A983;
019604  _DisposeDialog = $A983;
019605  _FindDItem = $A984;
019606  _Alert = $A985;
019607  _StopAlert = $A986;
```

```
019608  _NoteAlert = $A987;
019609  _CautionAlert = $A988;
019610  _CouldAlert = $A989;
019611  _FreeAlert = $A98A;
019612  _ParamText = $A98B;
019613  _ErrorSound = $A98C;
019614  _GetDItem = $A98D;
019615  _SetDItem = $A98E;
019616  _SetIText = $A98F;
019617  _GetIText = $A990;
019618  _ModalDialog = $A991;
019619  _DetachResource = $A992;
019620  _SetResPurge = $A993;
019621  _CurResFile = $A994;
019622  _InitResources = $A995;
019623  _RsrcZoneInit = $A996;
019624  _OpenResFile = $A997;
019625  _UseResFile = $A998;
019626  _UpdateResFile = $A999;
019627  _CloseResFile = $A99A;
019628  _SetResLoad = $A99B;
019629  _CountResources = $A99C;
019630  _GetIndResource = $A99D;
019631  _CountTypes = $A99E;
019632  _GetIndType = $A99F;
019633  _GetResource = $A9A0;
019634  _GetNamedResource = $A9A1;
019635  _LoadResource = $A9A2;
019636  _ReleaseResource = $A9A3;
019637  _HomeResFile = $A9A4;
019638  _SizeRsrc = $A9A5;
019639  _GetResAttrs = $A9A6;
019640  _SetResAttrs = $A9A7;
019641  _GetResInfo = $A9A8;
019642  _SetResInfo = $A9A9;
019643  _ChangedResource = $A9AA;
019644  _AddResource = $A9AB;
019645  _AddReference = $A9AC;
019646  _RmveResource = $A9AD;
019647  _RmveReference = $A9AE;
019648  _ResError = $A9AF;
019649  _WriteResource = $A9B0;
019650  _CreateResFile = $A9B1;
019651  _SystemEvent = $A9B2;
019652  _SystemClick = $A9B3;
019653  _SystemTask = $A9B4;
019654  _SystemMenu = $A9B5;
019655  _OpenDeskAcc = $A9B6;
019656  _CloseDeskAcc = $A9B7;
019657  _GetPattern = $A9B8;
019658  _GetCursor = $A9B9;
019659  _GetString = $A9BA;
019660  _GetIcon = $A9BB;
019661  _GetPicture = $A9BC;
019662  _GetNewWindow = $A9BD;
019663  _GetNewControl = $A9BE;
```

```
019664  _GetRMenu = $A9BF;
019665  _GetNewMBar = $A9C0;
019666  _UniqueID = $A9C1;
019667  _SysEdit = $A9C2;
019668  _OpenRFPerm = $A9C4;
019669  _RsrcMapEntry = $A9C5;
019670  _Secs2Date = $A9C6;
019671  _Date2Secs = $A9C7;
019672  _SysBeep = $A9C8;
019673  _SysError = $A9C9;
019674  _PutIcon = $A9CA;
019675  _Munger = $A9E0;
019676  _HandToHand = $A9E1;
019677  _PtrToXHand = $A9E2;
019678  _PtrToHand = $A9E3;
019679  _HandAndHand = $A9E4;
019680  _InitPack = $A9E5;
019681  _InitAllPacks = $A9E6;
019682  _Pack0 = $A9E7;
019683  _Pack1 = $A9E8;
019684  _Pack2 = $A9E9;
019685  _Pack3 = $A9EA;
019686  _FP68K = $A9EB;
019687  _Pack4 = $A9EB;
019688  _Elems68K = $A9EC;
019689  _Pack5 = $A9EC;
019690  _Pack6 = $A9ED;
019691  _DECSTR68K = $A9EE;
019692  _Pack7 = $A9EE;
019693  _PtrAndHand = $A9EF;
019694  _LoadSeg = $A9F0;
019695  _UnLoadSeg = $A9F1;
019696  _Launch = $A9F2;
019697  _Chain = $A9F3;
019698  _ExitToShell = $A9F4;
019699  _GetAppParms = $A9F5;
019700  _GetResFileAttrs = $A9F6;
019701  _SetResFileAttrs = $A9F7;
019702  _MethodDispatch = $A9F8;
019703  _InfoScrap = $A9F9;
019704  _UnlodeScrap = $A9FA;
019705  _UnloadScrap = $A9FA;
019706  _LodeScrap = $A9FB;
019707  _LoadScrap = $A9FB;
019708  _ZeroScrap = $A9FC;
019709  _GetScrap = $A9FD;
019710  _PutScrap = $A9FE;
019711  _Debugger = $A9FF;
019712  _IconDispatch = $ABC9;
019713  _DebugStr = $ABFF;
019714
019715  {
019716  ; Resource Manager
019717
019718   }
019719  _ResourceDispatch = $A822;
```

```
019720
019721  {
019722  ; PPCToolbox
019723
019724   }
019725  _PPC = $A0DD;
019726
019727  {
019728  ; Alias Manager
019729
019730   }
019731  _AliasDispatch = $A823;
019732
019733  {
019734  ; Component Manager
019735
019736   }
019737  _ComponentDispatch = $A82A;
019738
019739  {
019740  ; Device Manager (some shared by the File Manager)
019741
019742   }
019743  _Open = $A000;
019744  _Close = $A001;
019745  _Read = $A002;
019746  _Write = $A003;
019747  _Control = $A004;
019748  _Status = $A005;
019749  _KillIO = $A006;
019750
019751  {
019752  ; File Manager
019753
019754   }
019755  _GetVolInfo = $A007;
019756  _Create = $A008;
019757  _Delete = $A009;
019758  _OpenRF = $A00A;
019759  _Rename = $A00B;
019760  _GetFileInfo = $A00C;
019761  _SetFileInfo = $A00D;
019762  _UnmountVol = $A00E;
019763  _HUnmountVol = $A20E;
019764  _MountVol = $A00F;
019765  _Allocate = $A010;
019766  _GetEOF = $A011;
019767  _SetEOF = $A012;
019768  _FlushVol = $A013;
019769  _GetVol = $A014;
019770  _SetVol = $A015;
019771  _FInitQueue = $A016;
019772  _Eject = $A017;
019773  _GetFPos = $A018;
019774  _SetFilLock = $A041;
019775  _RstFilLock = $A042;
```

```
019776  _SetFilType = $A043;
019777  _SetFPos = $A044;
019778  _FlushFile = $A045;
019779  _HOpen = $A200;
019780  _HGetVInfo = $A207;
019781  _HCreate = $A208;
019782  _HDelete = $A209;
019783  _HOpenRF = $A20A;
019784  _HRename = $A20B;
019785  _HGetFileInfo = $A20C;
019786  _HSetFileInfo = $A20D;
019787  _AllocContig = $A210;
019788  _HSetVol = $A215;
019789  _HGetVol = $A214;
019790  _HSetFLock = $A241;
019791  _HRstFLock = $A242;
019792
019793  {
019794  ; dispatch trap for remaining File Manager (and Desktop Manager) calls
019795
019796   }
019797  _FSDispatch = $A060;
019798  _HFSDispatch = $A260;
019799
019800  {
019801  ; High level FSSpec calls
019802
019803   }
019804  _HighLevelFSDispatch = $AA52;
019805
019806  {
019807  ; Memory Manager
019808
019809   }
019810  _InitZone = $A019;
019811  _GetZone = $A11A;
019812  _SetZone = $A01B;
019813  _FreeMem = $A01C;
019814  _MaxMem = $A11D;
019815  _NewPtr = $A11E;
019816  _NewPtrSys = $A51E;
019817  _NewPtrClear = $A31E;
019818  _NewPtrSysClear = $A71E;
019819  _DisposPtr = $A01F;
019820  _DisposePtr = $A01F;
019821  _SetPtrSize = $A020;
019822  _GetPtrSize = $A021;
019823  _NewHandle = $A122;
019824  _NewHandleClear = $A322;
019825  _DisposHandle = $A023;
019826  _DisposeHandle = $A023;
019827  _SetHandleSize = $A024;
019828  _GetHandleSize = $A025;
019829  _HandleZone = $A126;
019830  _ReallocHandle = $A027;
019831  _RecoverHandle = $A128;
```

```
019832  _HLock = $A029;
019833  _HUnlock = $A02A;
019834  _EmptyHandle = $A02B;
019835  _InitApplZone = $A02C;
019836  _SetApplLimit = $A02D;
019837  _BlockMove = $A02E;
019838  _MemoryDispatch = $A05C;
019839  _MemoryDispatchA0Result = $A15C;
019840  _DeferUserFn = $A08F;
019841  _DebugUtil = $A08D;
019842
019843  {
019844  ; Event Manager
019845
019846   }
019847  _PostEvent = $A02F;
019848  _PPostEvent = $A12F;
019849  _OSEventAvail = $A030;
019850  _GetOSEvent = $A031;
019851  _FlushEvents = $A032;
019852  _VInstall = $A033;
019853  _VRemove = $A034;
019854  _OffLine = $A035;
019855  _MoreMasters = $A036;
019856  _WriteParam = $A038;
019857  _ReadDateTime = $A039;
019858  _SetDateTime = $A03A;
019859  _Delay = $A03B;
019860  _CmpString = $A03C;
019861  _DrvrInstall = $A03D;
019862  _DrvrRemove = $A03E;
019863  _InitUtil = $A03F;
019864  _ResrvMem = $A040;
019865  _GetTrapAddress = $A146;
019866  _SetTrapAddress = $A047;
019867  _GetOSTrapAddress = $A346;
019868  _SetOSTrapAddress = $A247;
019869  _GetToolTrapAddress = $A746;
019870  _SetToolTrapAddress = $A647;
019871  _GetToolBoxTrapAddress = $A746;
019872  _SetToolBoxTrapAddress = $A647;
019873  _PtrZone = $A148;
019874  _HPurge = $A049;
019875  _HNoPurge = $A04A;
019876  _SetGrowZone = $A04B;
019877  _CompactMem = $A04C;
019878  _PurgeMem = $A04D;
019879  _AddDrive = $A04E;
019880  _RDrvrInstall = $A04F;
019881  _LwrString = $A056;
019882  _UprString = $A054;
019883  _SetApplBase = $A057;
019884  _HWPriv = $A198;
019885
019886  {
019887  ; New names for (mostly) new flavors of old LwrString trap (redone <13>)
```

```
019888   }
019889  _LowerText = $A056;
019890  _StripText = $A256;
019891  _UpperText = $A456;
019892  _StripUpperText = $A656;
019893
019894  {
019895  ; Temporary Memory routines
019896
019897   }
019898  _OSDispatch = $A88F;
019899  _RelString = $A050;
019900  _ReadXPRam = $A051;
019901  _WriteXPRam = $A052;
019902  _InsTime = $A058;
019903  _InsXTime = $A458;
019904  _RmvTime = $A059;
019905  _PrimeTime = $A05A;
019906  _PowerOff = $A05B;
019907  _MaxBlock = $A061;
019908  _PurgeSpace = $A162;
019909  _MaxApplZone = $A063;
019910  _MoveHHi = $A064;
019911  _StackSpace = $A065;
019912  _NewEmptyHandle = $A166;
019913  _HSetRBit = $A067;
019914  _HClrRBit = $A068;
019915  _HGetState = $A069;
019916  _HSetState = $A06A;
019917  _InitFS = $A06C;
019918  _InitEvents = $A06D;
019919  _StripAddress = $A055;
019920  _Translate24To32 = $A091;
019921  _SetAppBase = $A057;
019922  _SwapMMUMode = $A05D;
019923  _SlotVInstall = $A06F;
019924  _SlotVRemove = $A070;
019925  _AttachVBL = $A071;
019926  _DoVBLTask = $A072;
019927  _SIntInstall = $A075;
019928  _SIntRemove = $A076;
019929  _CountADBs = $A077;
019930  _GetIndADB = $A078;
019931  _GetADBInfo = $A079;
019932  _SetADBInfo = $A07A;
019933  _ADBReInit = $A07B;
019934  _ADBOp = $A07C;
019935  _GetDefaultStartup = $A07D;
019936  _SetDefaultStartup = $A07E;
019937  _InternalWait = $A07F;
019938  _RGetResource = $A80C;
019939  _GetVideoDefault = $A080;
019940  _SetVideoDefault = $A081;
019941  _DTInstall = $A082;
019942  _SetOSDefault = $A083;
019943  _GetOSDefault = $A084;
```

```
019944  _IOPInfoAccess = $A086;
019945  _IOPMsgRequest = $A087;
019946  _IOPMoveData = $A088;
019947
019948  {
019949  ; Power Manager
019950
019951   }
019952  _PMgrOp = $A085;
019953  _IdleUpdate = $A285;
019954  _IdleState = $A485;
019955  _SerialPower = $A685;
019956  _Sleep = $A08A;
019957  _SleepQInstall = $A28A;
019958  _SlpQInstall = $A28A;
019959  _SleepQRemove = $A48A;
019960  _SlpQRemove = $A48A;
019961
019962  {
019963  ; Comm. Toolbox
019964
019965   }
019966  _CommToolboxDispatch = $A08B;
019967  _SysEnvirons = $A090;
019968
019969  {
019970  ; Egret Manager
019971
019972   }
019973  _EgretDispatch = $A092;
019974  _Gestalt = $A1AD;
019975  _NewGestalt = $A3AD;
019976  _ReplaceGestalt = $A5AD;
019977  _GetGestaltProcPtr = $A7AD;
019978  _InitProcMenu = $A808;
019979  _GetItemCmd = $A84E;
019980  _SetItemCmd = $A84F;
019981  _PopUpMenuSelect = $A80B;
019982  _KeyTrans = $A9C3;
019983
019984  {
019985  ; TextEdit
019986
019987   }
019988  _TEGetText = $A9CB;
019989  _TEInit = $A9CC;
019990  _TEDispose = $A9CD;
019991  _TextBox = $A9CE;
019992  _TESetText = $A9CF;
019993  _TECalText = $A9D0;
019994  _TESetSelect = $A9D1;
019995  _TENew = $A9D2;
019996  _TEUpdate = $A9D3;
019997  _TEClick = $A9D4;
019998  _TECopy = $A9D5;
019999  _TECut = $A9D6;
```

```
020000  _TEDelete = $A9D7;
020001  _TEActivate = $A9D8;
020002  _TEDeactivate = $A9D9;
020003  _TEIdle = $A9DA;
020004  _TEPaste = $A9DB;
020005  _TEKey = $A9DC;
020006  _TEScroll = $A9DD;
020007  _TEInsert = $A9DE;
020008  _TESetJust = $A9DF;
020009  _TEGetOffset = $A83C;
020010  _TEDispatch = $A83D;
020011  _TEStyleNew = $A83E;
020012
020013  {
020014  ; Color Quickdraw
020015
020016   }
020017  _OpenCPort = $AA00;
020018  _InitCPort = $AA01;
020019  _CloseCPort = $A87D;
020020  _NewPixMap = $AA03;
020021  _DisposPixMap = $AA04;
020022  _DisposePixMap = $AA04;
020023  _CopyPixMap = $AA05;
020024  _SetPortPix = $AA06;
020025  _NewPixPat = $AA07;
020026  _DisposPixPat = $AA08;
020027  _DisposePixPat = $AA08;
020028  _CopyPixPat = $AA09;
020029  _PenPixPat = $AA0A;
020030  _BackPixPat = $AA0B;
020031  _GetPixPat = $AA0C;
020032  _MakeRGBPat = $AA0D;
020033  _FillCRect = $AA0E;
020034  _FillCOval = $AA0F;
020035  _FillCRoundRect = $AA10;
020036  _FillCArc = $AA11;
020037  _FillCRgn = $AA12;
020038  _FillCPoly = $AA13;
020039  _RGBForeColor = $AA14;
020040  _RGBBackColor = $AA15;
020041  _SetCPixel = $AA16;
020042  _GetCPixel = $AA17;
020043  _GetCTable = $AA18;
020044  _GetForeColor = $AA19;
020045  _GetBackColor = $AA1A;
020046  _GetCCursor = $AA1B;
020047  _SetCCursor = $AA1C;
020048  _AllocCursor = $AA1D;
020049  _GetCIcon = $AA1E;
020050  _PlotCIcon = $AA1F;
020051  _OpenCPicture = $AA20;
020052  _OpColor = $AA21;
020053  _HiliteColor = $AA22;
020054  _CharExtra = $AA23;
020055  _DisposCTable = $AA24;
```

```
020056  _DisposeCTable = $AA24;
020057  _DisposCIcon = $AA25;
020058  _DisposeCIcon = $AA25;
020059  _DisposCCursor = $AA26;
020060  _DisposeCCursor = $AA26;
020061  _SeedCFill = $AA50;
020062  _CalcCMask = $AA4F;
020063  _CopyDeepMask = $AA51;
020064
020065  {
020066  ; Routines for video devices
020067
020068   }
020069  _GetMaxDevice = $AA27;
020070  _GetCTSeed = $AA28;
020071  _GetDeviceList = $AA29;
020072  _GetMainDevice = $AA2A;
020073  _GetNextDevice = $AA2B;
020074  _TestDeviceAttribute = $AA2C;
020075  _SetDeviceAttribute = $AA2D;
020076  _InitGDevice = $AA2E;
020077  _NewGDevice = $AA2F;
020078  _DisposGDevice = $AA30;
020079  _DisposeGDevice = $AA30;
020080  _SetGDevice = $AA31;
020081  _GetGDevice = $AA32;
020082  _DeviceLoop = $ABCA;
020083
020084  {
020085  ; Color Manager
020086
020087   }
020088  _Color2Index = $AA33;
020089  _Index2Color = $AA34;
020090  _InvertColor = $AA35;
020091  _RealColor = $AA36;
020092  _GetSubTable = $AA37;
020093  _UpdatePixMap = $AA38;
020094
020095  {
020096  ; Dialog Manager
020097
020098   }
020099  _NewCDialog = $AA4B;
020100  _MakeITable = $AA39;
020101  _AddSearch = $AA3A;
020102  _AddComp = $AA3B;
020103  _SetClientID = $AA3C;
020104  _ProtectEntry = $AA3D;
020105  _ReserveEntry = $AA3E;
020106  _SetEntries = $AA3F;
020107  _QDError = $AA40;
020108  _SaveEntries = $AA49;
020109  _RestoreEntries = $AA4A;
020110  _DelSearch = $AA4C;
020111  _DelComp = $AA4D;
```

```
020112  _SetStdCProcs = $AA4E;
020113  _StdOpcodeProc = $ABF8;
020114
020115  {
020116  ; added to Toolbox for color
020117
020118   }
020119  _SetWinColor = $AA41;
020120  _GetAuxWin = $AA42;
020121  _SetCtlColor = $AA43;
020122  _GetAuxCtl = $AA44;
020123  _NewCWindow = $AA45;
020124  _GetNewCWindow = $AA46;
020125  _SetDeskCPat = $AA47;
020126  _GetCWMgrPort = $AA48;
020127  _GetCVariant = $A809;
020128  _GetWVariant = $A80A;
020129
020130  {
020131  ; added to Menu Manager for color
020132
020133   }
020134  _DelMCEntries = $AA60;
020135  _GetMCInfo = $AA61;
020136  _SetMCInfo = $AA62;
020137  _DispMCInfo = $AA63;
020138  _GetMCEntry = $AA64;
020139  _SetMCEntries = $AA65;
020140
020141  {
020142  ; Menu Manager
020143
020144   }
020145  _MenuChoice = $AA66;
020146
020147  {
020148  ; Dialog Manager?
020149
020150   }
020151  _ModalDialogMenuSetup = $AA67;
020152  _DialogDispatch = $AA68;
020153
020154  {
020155  ; Font Manager
020156
020157   }
020158  _SetFractEnable = $A814;
020159  _FontDispatch = $A854;
020160
020161  {
020162  ; Palette Manager
020163
020164   }
020165  _InitPalettes = $AA90;
020166  _NewPalette = $AA91;
020167  _GetNewPalette = $AA92;
```

```
020168  _DisposePalette = $AA93;
020169  _ActivatePalette = $AA94;
020170  _SetPalette = $AA95;
020171  _NSetPalette = $AA95;
020172  _GetPalette = $AA96;
020173  _PmForeColor = $AA97;
020174  _PmBackColor = $AA98;
020175  _AnimateEntry = $AA99;
020176  _AnimatePalette = $AA9A;
020177  _GetEntryColor = $AA9B;
020178  _SetEntryColor = $AA9C;
020179  _GetEntryUsage = $AA9D;
020180  _SetEntryUsage = $AA9E;
020181  _CTab2Palette = $AA9F;
020182  _Palette2CTab = $AAA0;
020183  _CopyPalette = $AAA1;
020184  _PaletteDispatch = $AAA2;
020185
020186  {
020187  ; Sound Manager
020188
020189   }
020190  _SoundDispatch = $A800;
020191  _SndDisposeChannel = $A801;
020192  _SndAddModifier = $A802;
020193  _SndDoCommand = $A803;
020194  _SndDoImmediate = $A804;
020195  _SndPlay = $A805;
020196  _SndControl = $A806;
020197  _SndNewChannel = $A807;
020198  _SlotManager = $A06E;
020199  _ScriptUtil = $A8B5;
020200  _SCSIDispatch = $A815;
020201  _Long2Fix = $A83F;
020202  _Fix2Long = $A840;
020203  _Fix2Frac = $A841;
020204  _Frac2Fix = $A842;
020205  _Fix2X = $A843;
020206  _X2Fix = $A844;
020207  _Frac2X = $A845;
020208  _X2Frac = $A846;
020209  _FracCos = $A847;
020210  _FracSin = $A848;
020211  _FracSqrt = $A849;
020212  _FracMul = $A84A;
020213  _FracDiv = $A84B;
020214  _FixDiv = $A84D;
020215  _NMInstall = $A05E;
020216  _NMRemove = $A05F;
020217
020218  {
020219  ; All QDOffscreen Routines go through one trap with a selector
020220
020221   }
020222  _QDExtensions = $AB1D;
020223
```

```
020224  {
020225  ; UserDelay
020226
020227   }
020228  _UserDelay = $A84C;
020229  _InitDogCow = $A89F;
020230  _EnableDogCow = $A89F;
020231  _DisableDogCow = $A89F;
020232  _Moof = $A89F;
020233  _HFSPinaforeDispatch = $AA52;
020234
020235
020236
020237
020238
020239  {$ENDC} { UsingTraps }
020240
020241  {$IFC NOT UsingIncludes}
020242   END.
020243  {$ENDC}
020244
020245
020246  ### END OF FILE Traps.p
020247
```

```
020248
020249  ############################################################################
020250  ### FILE: Types.p
020251  ############################################################################
020252
020253
020254  {
020255  Created: Saturday, January 5, 1991 at 9:27 AM
020256   Types.p
020257   Pascal Interface to the Macintosh Libraries
020258
020259    Copyright Apple Computer, Inc. 1985-1991
020260    All rights reserved.
020261  }
020262
020263
020264  {$IFC UNDEFINED UsingIncludes}
020265  {$SETC UsingIncludes := 0}
020266  {$ENDC}
020267
020268  {$IFC NOT UsingIncludes}
020269   UNIT Types;
020270   INTERFACE
020271  {$ENDC}
020272
020273  {$IFC UNDEFINED UsingTypes}
020274  {$SETC UsingTypes := 1}
020275
020276  {$IFC UNDEFINED SystemSevenOrLater}
020277  {$SETC SystemSevenOrLater := FALSE}
020278  {$ENDC}
020279
020280  {$IFC UNDEFINED SystemSixOrLater}
020281  {$SETC SystemSixOrLater := SystemSevenOrLater}
020282  {$ENDC}
020283
020284
020285
020286  CONST
020287  noErr = 0;                  {All is well}
020288
020289
020290
020291
020292  TYPE
020293  Byte = 0..255;              { unsigned byte for fontmgr }
020294  SignedByte = - 128..127;    { any byte in memory }
020295  Ptr = ^SignedByte;
020296  Handle = ^Ptr;              {  pointer to a master pointer }
020297
020298  {$IFC UNDEFINED qMacApp}
020299  IntegerPtr = ^INTEGER;
020300  LongIntPtr = ^LONGINT;
020301  {$ENDC}
020302
020303  Fixed = LONGINT;            { fixed point arithmatic type }
```

```
020304  FixedPtr = ^Fixed;
020305  Fract = LONGINT;
020306  FractPtr = ^Fract;
020307  {$IFC OPTION(MC68881)}
020308  Extended80 = ARRAY [0..4] OF INTEGER;
020309  {$ELSEC}
020310  Extended80 = EXTENDED;
020311  {$ENDC}
020312
020313  VHSelect = (v,h);
020314
020315
020316  ProcPtr = Ptr;              { pointer to a procedure }
020317
020318  StringPtr = ^Str255;
020319  StringHandle = ^StringPtr;
020320
020321  Str255 = String[255];       { maximum string size }
020322
020323  Str63 = String[63];
020324
020325  Str32 = String[32];
020326
020327  Str31 = String[31];
020328
020329  Str27 = String[27];
020330
020331  Str15 = String[15];
020332
020333
020334
020335  OSErr = INTEGER;            { error code }
020336  OSType = PACKED ARRAY [1..4] OF CHAR;
020337  OSTypePtr = ^OSType;
020338  ResType = PACKED ARRAY [1..4] OF CHAR;
020339  ResTypePtr = ^ResType;
020340  ScriptCode = INTEGER;
020341  LangCode = INTEGER;
020342
020343
020344  PointPtr = ^Point;
020345  Point = RECORD
020346   CASE INTEGER OF
020347     1:
020348    (v: INTEGER;              {vertical coordinate}
020349    h: INTEGER);              {horizontal coordinate}
020350     2:
020351    (vh: ARRAY[VHSelect] OF INTEGER);
020352   END;
020353
020354  RectPtr = ^Rect;
020355  Rect = RECORD
020356   CASE INTEGER OF
020357     1:
020358    (top: INTEGER;
020359    left: INTEGER;
```

```
020360    bottom: INTEGER;
020361    right: INTEGER);
020362     2:
020363    (topLeft: Point;
020364    botRight: Point);
020365   END;
020366
020367
020368  PROCEDURE Debugger;
020369   INLINE $A9FF;
020370  PROCEDURE DebugStr(aStr: Str255);
020371   INLINE $ABFF;
020372  PROCEDURE SysBreak;
020373   INLINE $303C,$FE16,$A9C9;
020374  PROCEDURE SysBreakStr(debugStr: Str255);
020375   INLINE $303C,$FE15,$A9C9;
020376  PROCEDURE SysBreakFunc(debugFunc: Str255);
020377   INLINE $303C,$FE14,$A9C9;
020378
020379
020380
020381
020382  {$ENDC} { UsingTypes }
020383
020384  {$IFC NOT UsingIncludes}
020385   END.
020386  {$ENDC}
020387
020388
020389  ### END OF FILE Types.p
020390
```

```
020391
020392  ############################################################################
020393  ### FILE: Video.p
020394  ############################################################################
020395
020396  {
020397  Created: Sunday, January 6, 1991 at 11:26 PM
020398      Video.p
020399      Pascal Interface to the Macintosh Libraries
020400
020401          Copyright Apple Computer, Inc.    1986-1990
020402          All rights reserved
020403  }
020404
020405
020406  {$IFC UNDEFINED UsingIncludes}
020407  {$SETC UsingIncludes := 0}
020408  {$ENDC}
020409
020410  {$IFC NOT UsingIncludes}
020411      UNIT Video;
020412      INTERFACE
020413  {$ENDC}
020414
020415  {$IFC UNDEFINED UsingVideo}
020416  {$SETC UsingVideo := 1}
020417
020418  {$I+}
020419  {$SETC VideoIncludes := UsingIncludes}
020420  {$SETC UsingIncludes := 1}
020421  {$IFC UNDEFINED UsingQuickdraw}
020422  {$I $$Shell(PInterfaces)Quickdraw.p}
020423  {$ENDC}
020424  {$SETC UsingIncludes := VideoIncludes}
020425
020426  CONST
020427  mBaseOffset = 1;            {Id of mBaseOffset.}
020428  mRowBytes = 2;              {Video sResource parameter Id's }
020429  mBounds = 3;                {Video sResource parameter Id's }
020430  mVersion = 4;               {Video sResource parameter Id's }
020431  mHRes = 5;                  {Video sResource parameter Id's }
020432  mVRes = 6;                  {Video sResource parameter Id's }
020433  mPixelType = 7;             {Video sResource parameter Id's }
020434  mPixelSize = 8;             {Video sResource parameter Id's }
020435  mCmpCount = 9;              {Video sResource parameter Id's }
020436  mCmpSize = 10;              {Video sResource parameter Id's }
020437  mPlaneBytes = 11;           {Video sResource parameter Id's }
020438  mVertRefRate = 14;          {Video sResource parameter Id's }
020439  mVidParams = 1;             {Video parameter block id.}
020440  mTable = 2;                 {Offset to the table.}
020441  mPageCnt = 3;               {Number of pages}
020442  mDevType = 4;               {Device Type}
020443
020444  oneBitMode = 128;           {Id of OneBitMode Parameter list.}
020445  twoBitMode = 129;           {Id of TwoBitMode Parameter list.}
020446  fourBitMode = 130;          {Id of FourBitMode Parameter list.}
```

```
020447  eightBitMode = 131;         {Id of EightBitMode Parameter list.}
020448  sixteenBitMode = 132;       {Id of SixteenBitMode Parameter list.}
020449  thirtyTwoBitMode = 133;     {Id of ThirtyTwoBitMode Parameter list.}
020450
020451  firstVidMode = 128;         {The new, better way to do the above.    }
020452  secondVidMode = 129;        {    QuickDraw only supports six video    }
020453  thirdVidMode = 130;         {    at this time.                        }
020454  fourthVidMode = 131;
020455  fifthVidMode = 132;
020456  sixthVidMode = 133;
020457
020458  spGammaDir = 64;
020459  spVidNamesDir = 65;
020460
020461  { Control Codes }
020462  cscReset = 0;
020463  cscKillIO = 1;
020464  cscSetMode = 2;
020465  cscSetEntries = 3;
020466  cscSetGamma = 4;
020467  cscGrayPage = 5;
020468  cscGrayScreen = 5;
020469  cscSetGray = 6;
020470  cscSetInterrupt = 7;
020471  cscDirectSetEntries = 8;
020472  cscSetDefaultMode = 9;
020473
020474  { Status Codes }
020475  cscGetMode = 2;
020476  cscGetEntries = 3;
020477  cscGetPageCnt = 4;
020478  cscGetPages = 4;            { This is what C&D 2 calls it. }
020479  cscGetPageBase = 5;
020480  cscGetBaseAddr = 5;         { This is what C&D 2 calls it. }
020481  cscGetGray = 6;
020482  cscGetInterrupt = 7;
020483  cscGetGamma = 8;
020484  cscGetDefaultMode = 9;
020485
020486  TYPE
020487  VPBlockPtr = ^VPBlock;
020488  VPBlock = RECORD
020489      vpBaseOffset: LONGINT;  {Offset to page zero of video RAM (From minorBaseOS).}
020490      vpRowBytes: INTEGER;    {Width of each row of video memory.}
020491      vpBounds: Rect;         {BoundsRect for the video display (gives dimensions).}
020492      vpVersion: INTEGER;     {PixelMap version number.}
020493      vpPackType: INTEGER;
020494      vpPackSize: LONGINT;
020495      vpHRes: LONGINT;        {Horizontal resolution of the device (pixels per inch).}
020496      vpVRes: LONGINT;        {Vertical resolution of the device (pixels per inch).}
020497      vpPixelType: INTEGER;   {Defines the pixel type.}
020498      vpPixelSize: INTEGER;   {Number of bits in pixel.}
020499      vpCmpCount: INTEGER;    {Number of components in pixel.}
020500      vpCmpSize: INTEGER;     {Number of bits per component}
020501      vpPlaneBytes: LONGINT;  {Offset from one plane to the next.}
020502      END;
```

```
020503
020504  VDEntRecPtr = ^VDEntryRecord;
020505  VDEntryRecord = RECORD
020506      csTable: Ptr;           {(long) pointer to color table entry=value, r,g,b:INTEGER}
020507      END;
020508
020509  { Parm block for SetGray control call }
020510  VDGrayPtr = ^VDGrayRecord;
020511  VDGrayRecord = RECORD
020512      csMode: BOOLEAN;        {Same as GDDevType value (0=mono, 1=color)}
020513      END;
020514
020515  { Parm block for SetEntries control call }
020516  VDSetEntryPtr = ^VDSetEntryRecord;
020517  VDSetEntryRecord = RECORD
020518      csTable: ^ColorSpec;    {Pointer to an array of color specs}
020519      csStart: INTEGER;       {Which spec in array to start with, or -1}
020520      csCount: INTEGER;       {Number of color spec entries to set}
020521      END;
020522
020523  { Parm block for SetGamma control call }
020524  VDGamRecPtr = ^VDGammaRecord;
020525  VDGammaRecord = RECORD
020526      csGTable: Ptr;          {pointer to gamma table}
020527      END;
020528
020529  VDPgInfoPtr = ^VDPageInfo;
020530  VDPageInfo = RECORD
020531      csMode: INTEGER;        {(word) mode within device}
020532      csData: LONGINT;        {(long) data supplied by driver}
020533      csPage: INTEGER;        {(word) page to switch in}
020534      csBaseAddr: Ptr;        {(long) base address of page}
020535      END;
020536
020537  VDSzInfoPtr = ^VDSizeInfo;
020538  VDSizeInfo = RECORD
020539      csHSize: INTEGER;       {(word) desired/returned h size}
020540      csHPos: INTEGER;        {(word) desired/returned h position}
020541      csVSize: INTEGER;       {(word) desired/returned v size}
020542      csVPos: INTEGER;        {(word) desired/returned v position}
020543      END;
020544
020545  VDSettingsPtr = ^VDSettings;
020546  VDSettings = RECORD
020547      csParamCnt: INTEGER;    {(word) number of params}
020548      csBrightMax: INTEGER;   {(word) max brightness}
020549      csBrightDef: INTEGER;   {(word) default brightness}
020550      csBrightVal: INTEGER;   {(word) current brightness}
020551      csCntrstMax: INTEGER;   {(word) max contrast}
020552      csCntrstDef: INTEGER;   {(word) default contrast}
020553      csCntrstVal: INTEGER;   {(word) current contrast}
020554      csTintMax: INTEGER;     {(word) max tint}
020555      csTintDef: INTEGER;     {(word) default tint}
020556      csTintVal: INTEGER;     {(word) current tint}
020557      csHueMax: INTEGER;      {(word) max hue}
020558      csHueDef: INTEGER;      {(word) default hue}
```

```
020559      csHueVal: INTEGER;      {(word) current hue}
020560      csHorizDef: INTEGER;    {(word) default horizontal}
020561      csHorizVal: INTEGER;    {(word) current horizontal}
020562      csHorizMax: INTEGER;    {(word) max horizontal}
020563      csVertDef: INTEGER;     {(word) default vertical}
020564      csVertVal: INTEGER;     {(word) current vertical}
020565      csVertMax: INTEGER;     {(word) max vertical}
020566      END;
020567
020568
020569
020570  {$ENDC}    { UsingVideo }
020571
020572  {$IFC NOT UsingIncludes}
020573      END.
020574  {$ENDC}
020575
020576
020577  ### END OF FILE Video.p
020578
```

```
020579
020580  ############################################################################
020581  ### FILE: VideoIntf.p
020582  ############################################################################
020583
020584  {
020585      File: VideoIntf.p
020586
020587      As of MPW 3.0, interface files were reorganized to more closely
020588      match "Inside Macintosh" reference books and be more consistant
020589      from language to language.
020590
020591      Interfaces for the VideoIntf are now found in Video.p.
020592      This file, which includes Video.p, is provided for compatibility
020593      with old sources.
020594
020595      Pascal Interface to the Macintosh Libraries
020596      Copyright Apple Computer, Inc. 1988
020597      All Rights Reserved
020598  }
020599
020600  {$IFC UNDEFINED UsingIncludes}
020601  {$SETC UsingIncludes := 0}
020602  {$ENDC}
020603
020604  {$IFC NOT UsingIncludes}
020605      UNIT VideoIntf;
020606      INTERFACE
020607  {$ENDC}
020608
020609  {$IFC UNDEFINED UsingVideoIntf}
020610  {$SETC UsingVideoIntf := 1}
020611
020612  {$I+}
020613  {$SETC VideoIntfIncludes := UsingIncludes}
020614  {$SETC UsingIncludes := 1}
020615  {$IFC UNDEFINED UsingVideo}
020616  {$I $$Shell(PInterfaces)Video.p}
020617  {$ENDC}
020618  {$SETC UsingIncludes := VideoIntfIncludes}
020619
020620  {$ENDC}    { UsingVideoIntf }
020621
020622  {$IFC NOT UsingIncludes}
020623      END.
020624  {$ENDC}
020625
020626
020627  ### END OF FILE VideoIntf.p
020628
```

```
020629
020630  ##########################################################################
020631  ### FILE: Windows.p
020632  ##########################################################################
020633
020634  {
020635  Created: Saturday, December 15, 1990 at 8:08 PM
020636      Windows.p
020637      Pascal Interface to the Macintosh Libraries
020638
020639          Copyright Apple Computer, Inc.    1985-1990
020640          All rights reserved.
020641  }
020642
020643
020644  {$IFC UNDEFINED UsingIncludes}
020645  {$SETC UsingIncludes := 0}
020646  {$ENDC}
020647
020648  {$IFC NOT UsingIncludes}
020649      UNIT Windows;
020650      INTERFACE
020651  {$ENDC}
020652
020653  {$IFC UNDEFINED UsingWindows}
020654  {$SETC UsingWindows := 1}
020655
020656  {$I+}
020657  {$SETC WindowsIncludes := UsingIncludes}
020658  {$SETC UsingIncludes := 1}
020659  {$IFC UNDEFINED UsingQuickdraw}
020660  {$I $$Shell(PInterfaces)Quickdraw.p}
020661  {$ENDC}
020662  {$IFC UNDEFINED UsingEvents}
020663  {$I $$Shell(PInterfaces)Events.p}
020664  {$ENDC}
020665  {$IFC UNDEFINED UsingControls}
020666  {$I $$Shell(PInterfaces)Controls.p}
020667  {$ENDC}
020668  {$SETC UsingIncludes := WindowsIncludes}
020669
020670  CONST
020671  documentProc = 0;
020672  dBoxProc = 1;
020673  plainDBox = 2;
020674  altDBoxProc = 3;
020675  noGrowDocProc = 4;
020676  movableDBoxProc = 5;
020677  zoomDocProc = 8;
020678  zoomNoGrow = 12;
020679  rDocProc = 16;
020680  dialogKind = 2;
020681  userKind = 8;
020682
020683  {FindWindow Result Codes}
020684  inDesk = 0;
```

```
020685  inMenuBar = 1;
020686  inSysWindow = 2;
020687  inContent = 3;
020688  inDrag = 4;
020689  inGrow = 5;
020690  inGoAway = 6;
020691  inZoomIn = 7;
020692  inZoomOut = 8;
020693
020694  {window messages}
020695  wDraw = 0;
020696  wHit = 1;
020697  wCalcRgns = 2;
020698  wNew = 3;
020699  wDispose = 4;
020700  wGrow = 5;
020701  wDrawGIcon = 6;
020702
020703  {defProc hit test codes}
020704  wNoHit = 0;
020705  wInContent = 1;
020706  wInDrag = 2;
020707  wInGrow = 3;
020708  wInGoAway = 4;
020709  wInZoomIn = 5;
020710  wInZoomOut = 6;
020711  deskPatID = 16;
020712
020713  {Window Part Identifiers which correlate color table entries with window elements}
020714  wContentColor = 0;
020715  wFrameColor = 1;
020716  wTextColor = 2;
020717  wHiliteColor = 3;
020718  wTitleBarColor = 4;
020719
020720  TYPE
020721  WindowPeek = ^WindowRecord;
020722  WindowRecord = RECORD
020723      port: GrafPort;
020724      windowKind: INTEGER;
020725      visible: BOOLEAN;
020726      hilited: BOOLEAN;
020727      goAwayFlag: BOOLEAN;
020728      spareFlag: BOOLEAN;
020729      strucRgn: RgnHandle;
020730      contRgn: RgnHandle;
020731      updateRgn: RgnHandle;
020732      windowDefProc: Handle;
020733      dataHandle: Handle;
020734      titleHandle: StringHandle;
020735      titleWidth: INTEGER;
020736      controlList: ControlHandle;
020737      nextWindow: WindowPeek;
020738      windowPic: PicHandle;
020739      refCon: LONGINT;
020740      END;
```

```
020741
020742  CWindowPeek = ^CWindowRecord;
020743  CWindowRecord = RECORD
020744      port: CGrafPort;
020745      windowKind: INTEGER;
020746      visible: BOOLEAN;
020747      hilited: BOOLEAN;
020748      goAwayFlag: BOOLEAN;
020749      spareFlag: BOOLEAN;
020750      strucRgn: RgnHandle;
020751      contRgn: RgnHandle;
020752      updateRgn: RgnHandle;
020753      windowDefProc: Handle;
020754      dataHandle: Handle;
020755      titleHandle: StringHandle;
020756      titleWidth: INTEGER;
020757      controlList: ControlHandle;
020758      nextWindow: CWindowPeek;
020759      windowPic: PicHandle;
020760      refCon: LONGINT;
020761      END;
020762
020763  WStateDataPtr = ^WStateData;
020764  WStateDataHandle = ^WStateDataPtr;
020765  WStateData = RECORD
020766      userState: Rect;            {user state}
020767      stdState: Rect;             {standard state}
020768      END;
020769
020770  AuxWinPtr = ^AuxWinRec;
020771  AuxWinHandle = ^AuxWinPtr;
020772  AuxWinRec = RECORD
020773      awNext: AuxWinHandle;       {handle to next AuxWinRec}
020774      awOwner: WindowPtr;         {ptr to window }
020775      awCTable: CTabHandle;       {color table for this window}
020776      dialogCItem: Handle;        {handle to dialog manager structures}
020777      awFlags: LONGINT;           {reserved for expansion}
020778      awReserved: CTabHandle;     {reserved for expansion}
020779      awRefCon: LONGINT;          {user Constant}
020780      END;
020781
020782  WCTabPtr = ^WinCTab;
020783  WCTabHandle = ^WCTabPtr;
020784  WinCTab = RECORD
020785      wCSeed: LONGINT;            {reserved}
020786      wCReserved: INTEGER;        {reserved}
020787      ctSize: INTEGER;            {usually 4 for windows}
020788      ctTable: ARRAY [0..4] OF ColorSpec;
020789      END;
020790
020791
020792  PROCEDURE InitWindows;
020793      INLINE $A912;
020794  PROCEDURE GetWMgrPort(VAR wPort: GrafPtr);
020795      INLINE $A910;
020796  FUNCTION NewWindow(wStorage: Ptr;boundsRect: Rect;title: Str255;visible: BOOLEAN;
```

```
020797      theProc: INTEGER;behind: WindowPtr;goAwayFlag: BOOLEAN;refCon: LONGINT): WindowPtr;
020798      INLINE $A913;
020799  FUNCTION GetNewWindow(windowID: INTEGER;wStorage: Ptr;behind: WindowPtr): WindowPtr;
020800      INLINE $A9BD;
020801  PROCEDURE CloseWindow(theWindow: WindowPtr);
020802      INLINE $A92D;
020803  PROCEDURE DisposeWindow(theWindow: WindowPtr);
020804      INLINE $A914;
020805  PROCEDURE GetWTitle(theWindow: WindowPtr;VAR title: Str255);
020806      INLINE $A919;
020807  PROCEDURE SelectWindow(theWindow: WindowPtr);
020808      INLINE $A91F;
020809  PROCEDURE HideWindow(theWindow: WindowPtr);
020810      INLINE $A916;
020811  PROCEDURE ShowWindow(theWindow: WindowPtr);
020812      INLINE $A915;
020813  PROCEDURE ShowHide(theWindow: WindowPtr;showFlag: BOOLEAN);
020814      INLINE $A908;
020815  PROCEDURE HiliteWindow(theWindow: WindowPtr;fHilite: BOOLEAN);
020816      INLINE $A91C;
020817  PROCEDURE BringToFront(theWindow: WindowPtr);
020818      INLINE $A920;
020819  PROCEDURE SendBehind(theWindow: WindowPtr;behindWindow: WindowPtr);
020820      INLINE $A921;
020821  FUNCTION FrontWindow: WindowPtr;
020822      INLINE $A924;
020823  PROCEDURE DrawGrowIcon(theWindow: WindowPtr);
020824      INLINE $A904;
020825  PROCEDURE MoveWindow(theWindow: WindowPtr;hGlobal: INTEGER;vGlobal: INTEGER;
020826      front: BOOLEAN);
020827      INLINE $A91B;
020828  PROCEDURE SizeWindow(theWindow: WindowPtr;w: INTEGER;h: INTEGER;fUpdate: BOOLEAN);
020829      INLINE $A91D;
020830  PROCEDURE ZoomWindow(theWindow: WindowPtr;partCode: INTEGER;front: BOOLEAN);
020831      INLINE $A83A;
020832  PROCEDURE InvalRect(badRect: Rect);
020833      INLINE $A928;
020834  PROCEDURE InvalRgn(badRgn: RgnHandle);
020835      INLINE $A927;
020836  PROCEDURE ValidRect(goodRect: Rect);
020837      INLINE $A92A;
020838  PROCEDURE ValidRgn(goodRgn: RgnHandle);
020839      INLINE $A929;
020840  PROCEDURE BeginUpdate(theWindow: WindowPtr);
020841      INLINE $A922;
020842  PROCEDURE EndUpdate(theWindow: WindowPtr);
020843      INLINE $A923;
020844  PROCEDURE SetWRefCon(theWindow: WindowPtr;data: LONGINT);
020845      INLINE $A918;
020846  FUNCTION GetWRefCon(theWindow: WindowPtr): LONGINT;
020847      INLINE $A917;
020848  PROCEDURE SetWindowPic(theWindow: WindowPtr;pic: PicHandle);
020849      INLINE $A92E;
020850  FUNCTION GetWindowPic(theWindow: WindowPtr): PicHandle;
020851      INLINE $A92F;
020852  FUNCTION CheckUpdate(VAR theEvent: EventRecord): BOOLEAN;
```

```
020853      INLINE $A911;
020854  PROCEDURE ClipAbove(window: WindowPeek);
020855      INLINE $A90B;
020856  PROCEDURE SaveOld(window: WindowPeek);
020857      INLINE $A90E;
020858  PROCEDURE DrawNew(window: WindowPeek;update: BOOLEAN);
020859      INLINE $A90F;
020860  PROCEDURE PaintOne(window: WindowPeek;clobberedRgn: RgnHandle);
020861      INLINE $A90C;
020862  PROCEDURE PaintBehind(startWindow: WindowPeek;clobberedRgn: RgnHandle);
020863      INLINE $A90D;
020864  PROCEDURE CalcVis(window: WindowPeek);
020865      INLINE $A909;
020866  PROCEDURE CalcVisBehind(startWindow: WindowPeek;clobberedRgn: RgnHandle);
020867      INLINE $A90A;
020868  FUNCTION GrowWindow(theWindow: WindowPtr;startPt: Point;bBox: Rect): LONGINT;
020869      INLINE $A92B;
020870  FUNCTION FindWindow(thePoint: Point;VAR theWindow: WindowPtr): INTEGER;
020871      INLINE $A92C;
020872  FUNCTION PinRect(theRect: Rect;thePt: Point): LONGINT;
020873      INLINE $A94E;
020874  FUNCTION DragGrayRgn(theRgn: RgnHandle;startPt: Point;boundsRect: Rect;
020875      slopRect: Rect;axis: INTEGER;actionProc: ProcPtr): LONGINT;
020876      INLINE $A905;
020877  FUNCTION TrackBox(theWindow: WindowPtr;thePt: Point;partCode: INTEGER): BOOLEAN;
020878      INLINE $A83B;
020879  PROCEDURE GetCWMgrPort(VAR wMgrCPort: CGrafPtr);
020880      INLINE $AA48;
020881  PROCEDURE SetWinColor(theWindow: WindowPtr;newColorTable: WCTabHandle);
020882      INLINE $AA41;
020883  FUNCTION GetAuxWin(theWindow: WindowPtr;VAR awHndl: AuxWinHandle): BOOLEAN;
020884      INLINE $AA42;
020885  PROCEDURE SetDeskCPat(deskPixPat: PixPatHandle);
020886      INLINE $AA47;
020887  FUNCTION NewCWindow(wStorage: Ptr;boundsRect: Rect;title: Str255;visible: BOOLEAN;
020888      procID: INTEGER;behind: WindowPtr;goAwayFlag: BOOLEAN;refCon: LONGINT): WindowPtr;
020889      INLINE $AA45;
020890  FUNCTION GetNewCWindow(windowID: INTEGER;wStorage: Ptr;behind: WindowPtr): WindowPtr;
020891      INLINE $AA46;
020892  FUNCTION GetWVariant(theWindow: WindowPtr): INTEGER;
020893      INLINE $A80A;
020894  FUNCTION GetGrayRgn: RgnHandle;
020895      INLINE $2EB8,$09EE;
020896  PROCEDURE SetWTitle(theWindow: WindowPtr;title: Str255);
020897      INLINE $A91A;
020898  FUNCTION TrackGoAway(theWindow: WindowPtr;thePt: Point): BOOLEAN;
020899      INLINE $A91E;
020900  PROCEDURE DragWindow(theWindow: WindowPtr;startPt: Point;boundsRect: Rect);
020901      INLINE $A925;
020902
020903
020904  {$ENDC}    { UsingWindows }
020905
020906  {$IFC NOT UsingIncludes}
020907      END.
020908  {$ENDC}
```

```
020909
020910
020911  ### END OF FILE Windows.p
```