AN220

Watt-Hour Meter using PIC16C923 and CS5460

Authors: Brett Duane, Stephen Humberd
Microchip Technology Inc.

OVERVIEW

This application note shows how to use a PIC16C923 microcontroller to control operation of the CS5460 power measurement integrated circuit from Cirrus Logic®/Crystal Power Measurement, to drive a liquid crystal panel (“glass”), and to store and retrieve data using the 24C01 Serial EEPROM.

Energy transferred between the line and load is measured by the CS5460. The PIC16C923 initializes the CS5460 with calibration data stored in the 24C01 Serial EEPROM, records the total energy measured in the 24C01, and displays results on the LCD panel.

INTRODUCTION

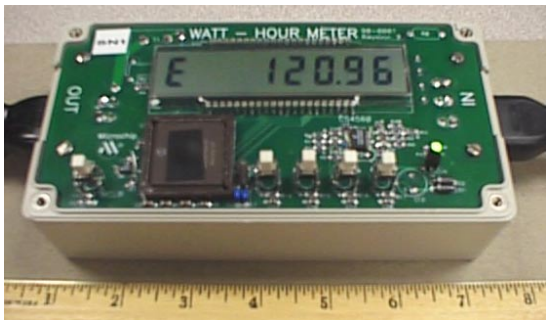
Most forms of AC power measurement have already been patented by various companies, so measuring AC power in a product intended for sale often involves paying licensing fees to another company. The CS5460 offers an integrated solution that provides a power and energy measurement sub-system, requiring only voltage and current sense inputs. In addition, calibration is accurate for any current waveform or power factor that may be encountered.

By using the CS5460, a PIC16C923 microcontroller, a 24C01 Serial EEPROM, and an LCD panel, a simple and compact device is constructed that displays RMS voltage, RMS current, and the energy consumed by a load. These features are extended by including computation and display of apparent power, true power, and power factor.

The PIC16C923 LCD controller can drive an LCD panel with up to 4 common planes and up to 32 segments. 4K words of program memory, and 176 bytes of RAM are provided. A Synchronous Serial Peripheral (SSP) provides SPI™ communications with the CS5460. Inter-Integrated Circuit™ (I^2C) communications with the 24C01 Serial EEPROM are provided by firmware.

The CS5460 power/energy measurement IC measures instantaneous voltage and current four thousand times a second and uses these measurements to compute V_{RMS}, I_{RMS}, instantaneous power, and accumulated energy results for read out. In addition, a pulse is generated whenever a user specified amount of energy transfers between the line and the load.

FIGURE 1: WATT-HOUR METER



HARDWARE

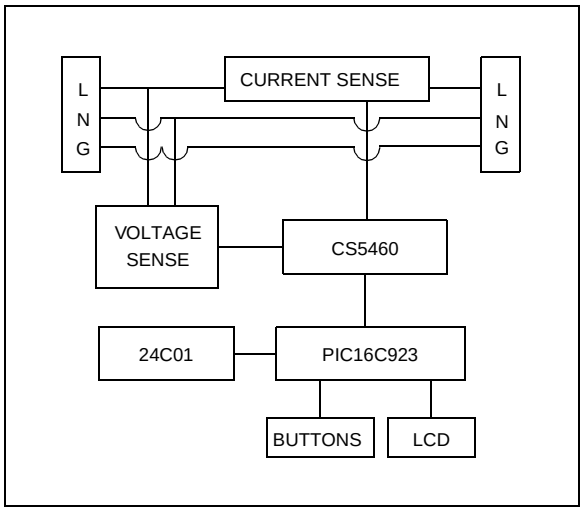
On power-up, the PIC16C923 microcontroller reads the calibration data, device serial number, and total energy from the 24C01, writes the calibration data to the CS5460, initializes the CS5460, and reads the state of the control buttons.

If the control button state matches one of three patterns at RESET, a control mode is entered that allows setting the real time clock (RTC), clearing the total WHr and restoring default calibration values, or adjusting calibration constants.

During normal operation, the PIC16C923 counts pulses from the CS5460, reads CS5460 data registers, drives the LCD panel to display the requested data, and monitors the control buttons. The pulses are used to update the total energy count and are periodically written to the 24C01.

The CS5460 measures line voltage and line current to compute power and energy transferred on the line. When a unit of energy has transferred between the line and the load, a pulse with direction indication is generated.

FIGURE 2: SYSTEM BLOCK DIAGRAM



CS5460 Power/Energy Measurement Circuit

The CS5460 measures the instantaneous line voltage and line current, four thousand times a second. These measurements are used to compute instantaneous power, energy transferred since the last measurement, RMS voltage, RMS current, and accumulated energy transferred. All measurements and results can be read by an external controller, via the SPI interface. A transfer of energy is also indicated by a pulse output at the $\overline{\text{EOUT}}$ pin. The direction of transfer is indicated by the $\overline{\text{EDIR}}$ pin.

Communication with the CS5460 takes place over a 4-wire SPI link with the PIC16C923. The CS5460 is configured and controlled over this link. Calculation results are also read by the controller over this link.

The line voltage may be sampled using a transformer or resistor divider. The differential input is limited to 150mV$_{RMS}$. In this application, line voltage is detected from the secondary winding of the power supply transformer, T2 (see Figure 3B and Figure 5). When operating from 120V, there is about an 8V peak at V$_{IN+}$ or V$_{IN-}$. When operating from 220V, there is about a 14.7V peak. This voltage is further reduced by a resistor network before being applied to the CS5460 (see Figure 4A).

The line current may be sampled using a current transformer or shunt resistor. Depending on the gain of the input channel, the differential input is limited to either 30 mV$_{RMS}$ (gain = 50), or 150 mV$_{RMS}$ (gain = 10). In this application, the current channel gain is 10, for a maximum input voltage of 150 mV$_{RMS}$. This voltage is provided by the current sense transformer T1 and resistor R21, and is reduced by a resistor network similar to the line voltage channel (see Figure 3A and Figure 4B).

There is no switching provided, or required for operation from either 120V or 220V, 50Hz or 60Hz. However, accuracy will decrease when operating from a line voltage different than the calibration conditions.

By using the instantaneous voltage and current, the CS5460 computes the RMS voltage, RMS current, and instantaneous power. The instantaneous power is integrated at the sampling rate (4000Hz) to compute the energy transferred. A new RMS value is available every 4000 samples. Samples are taken 4000 times per second, or about 67 times per 60Hz cycle.

When the integrated energy exceeds 10 WSec, a fixed width pulse is generated at the $\overline{\text{EOUT}}$ pin and the integrated energy is reduced by 10 WSec. These pulses are counted to record energy consumption. The $\overline{\text{EDIR}}$ pin indicates the direction that the energy flows (reactive loads can return energy to the line). Depending on the state of the $\overline{\text{EDIR}}$ pin, the pulse at the $\overline{\text{EOUT}}$ pin causes the PIC16C923 to either increment, or decrement the total energy count.

FIGURE 3: CIRCUITS THAT MONITOR LINE CURRENT (A) AND LINE VOLTAGE (B)

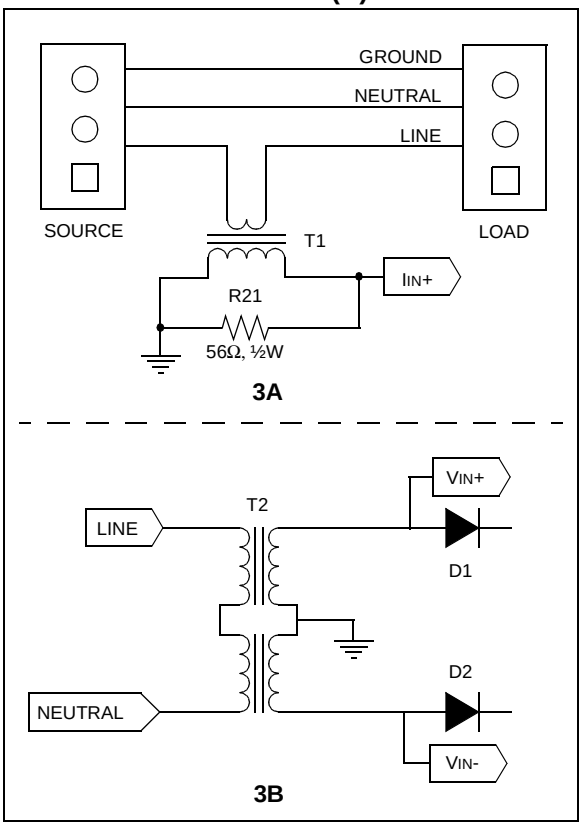
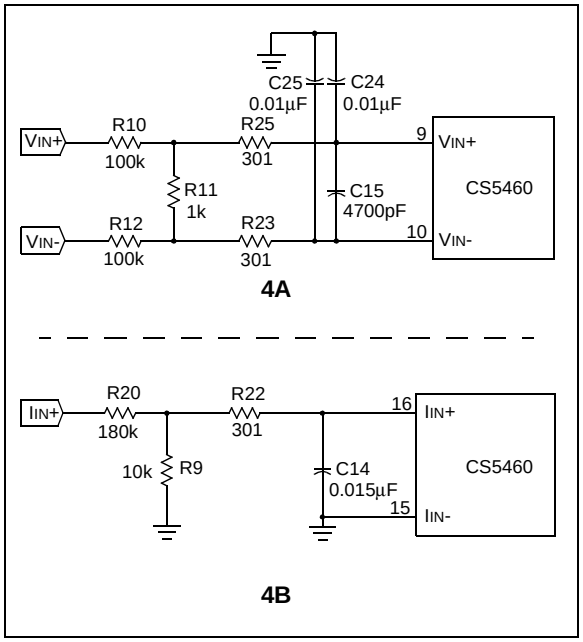


FIGURE 4: CS5460 INPUT ATTENUATION CIRCUITS

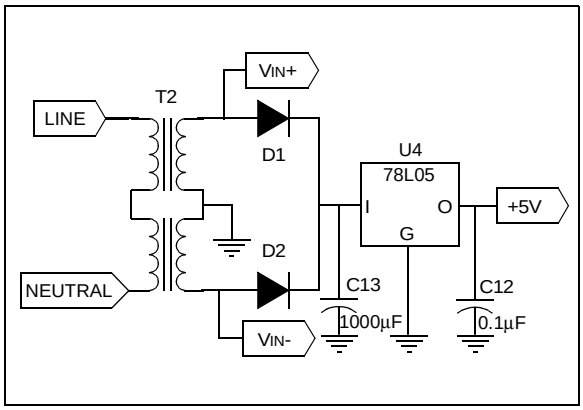


Power Supply

A transformer isolated power supply provides power for the Watt-Hour Meter. The transformer primary is connected to the line between the power source and the current sense transformer. The AC voltage from the transformer secondary is used to detect the line voltage and is coupled to the CS5460 through a resistor network (see Figure 4A).

The AC from the center tapped secondary is full wave rectified, filtered, and provided to the 5V regulator. The 5V loads are the “power-on” LED, the CS5460, and the PIC16C923. The majority of the current is drawn by the LED, about 7.5mA. The rest of the circuit draws less than 5mA.

FIGURE 5: POWER SUPPLY CIRCUIT

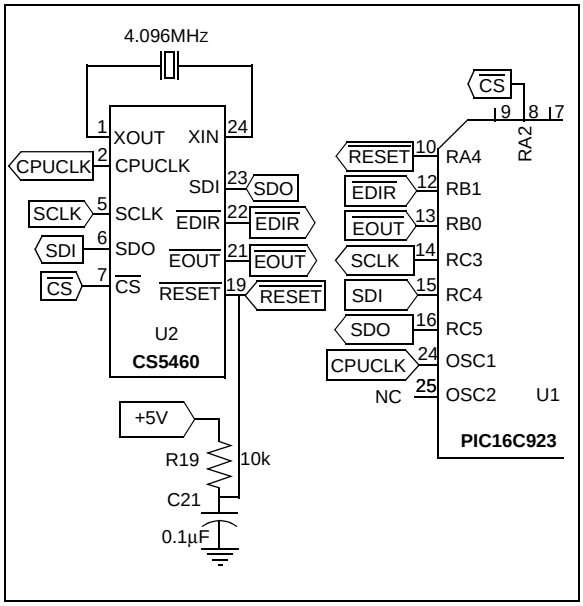


PIC16C923 Microcontroller

The PIC16C923 microcontroller provides a Liquid Crystal Display (LCD) driver module that drives the LCD panel directly. It also communicates with the CS5460 using the 4-wire SPI link (SDI, SDO, SCL, and $\overline{CS}$) to issue commands, write calibration data, and read measurement and calculation results. The microcontroller also controls the CS5460 $\overline{RESET}$ line (see Figure 6).

The controller system oscillator is driven by the CPUCLK output of the CS5460 and operates at 4.096MHz. The system oscillator is configured for XT mode, but any crystal mode will work. A 32.768kHz crystal has been provided for use with the Timer1 oscillator. Since the CS5460 provides a 4.096MHz clock source to the PIC16C923, either source can be used for the real time clock source. The demonstration units have been configured to use the CS5460 clock source for the real time clock.

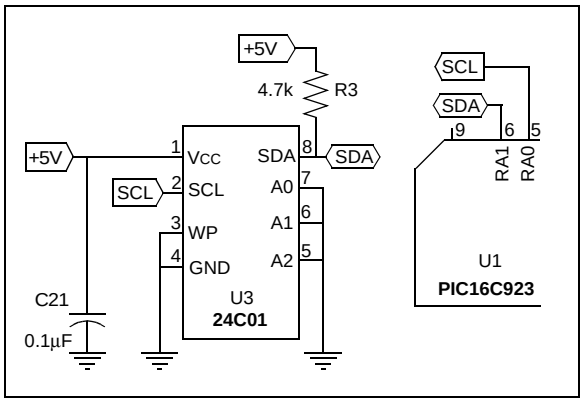
FIGURE 6: CONNECTIONS BETWEEN THE CS5460 AND THE PIC16C923



Interface to 24C01 Serial EEPROM

The Serial EEPROM stores the calibration constants required by the CS5460 for accurate measurements, and the total accumulated energy transferred. The controller communicates with the 24C01 via an I^2C interface. Since the SSP module is already in use supporting SPI communications with the CS5460, the PIC16C923 must perform I^2C communications in firmware, using RA0 (SCL) and RA1 (SDA) (see Figure 7). Either the 24C01, or the PIC16C923, may pull the SDA line low, depending on the direction of the data flow. Since the PIC16C923 always drives the SCL line, no pull-up resistor was included. A memory map of the Serial EEPROM is included in Appendix A.

FIGURE 7: CONNECTION BETWEEN THE 24C01 AND THE PIC16C923



User Interface

The user interface consists of the LCD display, four control push buttons, one reset push button, and the “power-on” LED.

The PIC16C923 LCD module directly drives the LCD panel. The panel can display eight, 7-segment digits (numbers only), seven decimal points and three colons.

When pushed, each of the four push buttons pull the respective port pin low. The buttons are connected to PORTB and are numbered from 1 to 4, left to right.

FIRMWARE

The CS5460 transfers data in 4 byte groups (32-bits). The first byte contains the register address and a bit specifying a read or write operation. The remaining 3 bytes are transferred to or from one of the internal registers. The CS5460 also accepts single byte commands. Such commands are followed by 3 SYNC bytes that are treated as `NOP` bytes.

A write command is followed by 3 bytes of data to the CS5460, to be written to the selected register. A read command causes the 3 bytes of the selected register to be output by the CS5460.

If the command byte specifies an operation to be performed, or a read operation, the remaining 3 bytes transmitted by the PIC16C923 should be SYNC0 bytes (0xFE).

Power-up and RESET

A Power-on Reset initializes the CS5460 and clears the real time clock.

Initialize On-Chip Peripherals

Timer1, Timer2, the SSP, and Ports A, B, and C are configured for operation. Interrupts are also enabled. The LCD module is then configured.

Clear the LCD Display

All segment data registers are cleared to blank the display. This routine is called frequently during normal operation.

Initialize Variables

If this was a cold start (power has just been applied), the memory contents are cleared and the calibration constants are copied from the 24C01 to the CS5460. The device serial number and the current total WHr are retrieved from the 24C01. If a warm start has occurred, only the serial number is retrieved.

Initialize the CS5460

The CS5460 is configured to generate a pulse at the $\overline{EOUT}$ pin for each 10 WSec measured (360 pulses per WHr). For 100W loads, this causes 10 pulses per second to be generated.

Check Button Status

The status of the four push buttons is checked. If all four buttons are pressed, the total WHr value in the 24C01 is cleared and calibration values are copied from the EEPROM to the CS5460. If the center two buttons are pressed, the real time clock is set. If the outer two buttons are pressed, the Watt-Hour meter enters Calibration mode (see Table 1). If no buttons are pressed, the CS5460 begins continuous measurements. Execution proceeds to the scrolling start-up message.

TABLE 1: BUTTON STATES CHECKED DURING RESET

Buttons				Control Mode
1	2	3	4	
X	X	X	X	Clear WHr, restore calibration values
	X	X		Set Clock
X			X	Calibration

Display the Start-up Message

A start-up message is displayed on the LCD. This message scrolls across the display until any of the four buttons is pressed. This message displays the device name and serial number.

Normal Operation

Results of the various calculations are displayed on the LCD. Each result is displayed for two seconds with an update after one second. If no buttons are pressed, the next mode is displayed. Holding any button keeps the display in the present mode, for as long as the button is held. New results will be displayed each second (see Table 2).

TABLE 2: DISPLAY MODES

Display	Value Displayed
HH:MM:SS	Time
E	RMS Voltage
C	RMS Current
AP	Apparent Power
TP	True Power
PF	Power Factor
Hr	Total WHr

Time

The first result displayed is the time of day in the form HH:MM:SS. If the time of day was not set at RESET, this indicates the time since power-up (days are not recorded).

RMS Voltage

The RMS voltage is computed by reading the RMS voltage value from the CS5460. This is a 24-bit value with a range of 0.000 to 1.000, representing a fraction of the full scale voltage. The 16 most significant bits are multiplied by the full scale voltage (as 16-bits) to produce the actual RMS voltage on the line, as a 16-bit binary number. This is converted to a 5 digit packed BCD number. The LCD display is blanked and "E" and the appropriate decimal point are displayed. The packed BCD number is then displayed, after determining which digits (leading zeros) should remain blank.

After one second, the value is updated and displayed again. After another second, execution proceeds to the next mode if no buttons were pressed. If a button was pressed, the two-second counter that controls when the next subroutine should be executed is cleared, extending the time that the value is displayed. This code is repeated in all display subroutines.

RMS Current

The RMS current is computed and displayed similar to the RMS voltage. The only differences are the full scale current is used, a "C" is displayed and a different decimal point is turned on. The button state is again checked to see if execution should remain in this subroutine.

Apparent Power

The apparent power is computed in a subroutine (`CalcAP`), called by both the apparent power loop (`APLoop`) and the power factor loop (`PFLoop`). The apparent power is computed by reading the RMS voltage and RMS current from the CS5460, as before. The 16 most significant bits of each are multiplied together, giving a 32-bit result. The 16 most significant bits of the result are multiplied with the full scale apparent power (16-bits) to get the actual apparent power in volt • amps in binary. The 16 most significant bits are returned for use by the calling subroutines.

`APLoop` then converts the apparent power in binary (16-bits) to a 5-digit packed BCD number for display. The LCD display is blanked, "AP" is displayed and after determining which digits should remain blanked (leading zeros), the apparent power is displayed. The buttons are again checked, as before, to determine if execution should remain in this subroutine.

True Power

The CS5460 was programmed to generate a pulse whenever 10 WSec of energy has been transferred. For a 250W load, 25 pulses each second will be generated. These pulses have been processed in an Interrupt Service Routine. When 360 pulses have been accumulated, 3600 WSec or 1 WHr has been transferred. The total WHr is then incremented. The pulse count is also recorded for each second.

The apparent power is computed in a subroutine (`CalcTP`), called by both the true power loop (`TPLoop`) and the power factor loop (`PFLoop`). `CalcTP` multiplies the number of pulses received during the last second by 10 to compute the true power consumed by the load. The result is returned as the true power in watts as a 16-bit binary number.

`TPLoop` converts and displays the true power in the same way as `APLoop` displays apparent power. The only difference is that "TP" is displayed instead of "AP". The buttons are again checked as before to determine if execution should remain in this subroutine.

Power Factor (PF)

The power factor is computed by calling the `CalcAP` subroutine to get apparent power in volt • amps and the `CalcTP` subroutine to get true power in watts. The true power is divided by the apparent power to get the power factor as a binary result, in the range of 0.000 to 1.000. The binary power factor is multiplied by 1000 and converted to a BCD number for display. The appropriate decimal point is turned on.

The buttons are checked as before to determine if execution should remain in the power factor subroutine.

If apparent power is equal to 0, there is no load (I_{RMS} = 0). This can result in a division by zero condition. If a division by zero is detected, a PF of 1.000 is reported.

The calculated power factor can also greatly exceed 1.0. When this occurs, the power factor is reported as being 1.000. This occurs when the load characteristics are rapidly changing (as when a motor is starting). All the measurements are not taken at exactly the same time and the RMS values are calculated over a one second period. When the load reaches a steady state condition, the power factor will again be correct.

Energy (Watt-Hours)

The `WHLoop` simply displays the WHr counter value. The binary count is converted to BCD. The BCD number is then displayed with leading zeros blanked, using the same subroutine used by the apparent power and true power displays. The WHr counter is 16-bits long, allowing a maximum of 65,535 WHr to be displayed. When this count is exceeded, the count rolls over to 0.

Control Modes

If any of the three control modes was selected during RESET, execution branches to one of these modules to control how the Watt-Hour meter functions.

When the control mode is terminated, a warm start RESET is executed.

Calibrating the Watt-Hour Meter

The user is given the opportunity to adjust the calibration constants. These constants will not be stored to the 24C01. When reset, using the reset button, the calibration values entered in the Calibrate mode will be used for making measurements and operation will resume as normal, except that the new calibration values are used. If reset by removing power, or reset while pressing all four buttons (clear total WHr), the constants stored in the 24C01 will be used for operation.

Enter Calibration mode by holding the two outer buttons while pressing the reset button. The first three digits display “CAL”, and the remaining digits indicate which constant is being adjusted. “CAL EOFF” will be displayed first. This indicates that the value displayed the next time button 2 is pressed, will be the calibration constant for the voltage offset. Pressing button 2 again displays the constant’s value.

The decimal point is next to the digit to be modified. Pressing button 3 will move the decimal point to the next digit to the right. Pressing button 4 will increment that digit. Each digit will cycle from 0 to F, then back to 0. Only that digit will be affected. Pressing button 1 at any time will cause the value for that constant to be sent to the CS5460 and display the next constant name. See Table 3 for button functions.

TABLE 3: CALIBRATION MODE BUTTON FUNCTIONS

Button				Function
1	2	3	4	
X				Writes constant to CS5460 and displays next constant name
	X			Displays each constant name and its value in turn
		X		Selects next digit and moves decimal point
			X	Increments selected digit

Table 4 shows the constant names and typical values. It is essential that the offsets be minimized before setting the gains.

To set the offsets, remove AC power from the Watt-Hour meter and apply DC power of 8 to 12 V_{DC} to C13. Adjust the offset constants for minimum RMS results (there is a null in both the current and voltage channels). Record the offsets.

Apply AC power to the Watt-Hour meter and remove the DC power from C13. Applying power in this order prevents a loss of power to the CS5460. If power is lost, reenter the offset values before adjusting the gain values. Apply a known resistive load to the Watt-Hour meter output. Adjust the voltage and current gain constants so the indicated RMS voltage and current match the actual load voltage and current. Adjust the pulse rate gain so the indicated true power matches the actual load power. Record the gain constants.

Resetting the device now uses the constants just found. If the total Watt-Hours is cleared, the original constants will be restored.

The software was designed for demonstration purposes; therefore, the calibration constants cannot be written to the serial EEPROM. If desired, the user can modify the code to write the new calibration constants to the EEPROM.

TABLE 4: CALIBRATION MODE INDICATIONS, CONSTANT AND TYPICAL VALUES

Indication	Constant	Calibration Value (120V, 10A)	CS5460 Default
EOFF	Voltage Offset	0x00CCBB = +0.00624	0x000000 = 0.00000
COFF	Current Offset	0xFEB320 = -0.01015	0x000000 = 0.00000
E GA	Voltage Gain	0x2C2F62 = 0.69039	0x400000 = 1.00000
C GA	Current Gain	0x298610 = 0.64917	0x400000 = 1.00000
P GA	Pulse Rate Gain	0x01FEF2 = 510.95	0x0FA000 = 4000.000

Clear Total Watt-Hours

This option causes the total WHr to be cleared from the 24C01 and RAM, and copies calibration data stored in the 24C01 back to the CS5460. The word “CLEAR” is displayed until the buttons are released.

Setting the Real Time Clock

“CL” is displayed in the two digits at the left edge of the display. The current time is displayed in the remaining six digits.

If buttons 2 and 4 are pressed together, the hours are incremented. If buttons 3 and 4 are pressed, the minutes are incremented. If the minutes roll over from 59 to 0, the hours will not be affected. If button 1 is pressed, “CL” is cleared from the display and execution proceeds to the main loop. Pressing Button 2, 3, or 4 alone has no effect (see Table 5).

TABLE 5: CLOCK SET MODE BUTTON FUNCTIONS

Button				Function
1	2	3	4	
X				Done setting clock
	X		X	Increment hours
		X	X	Increment minutes

POSSIBLE ENHANCEMENTS

An idea to simplify the calibration process is presented, along with ideas for adding a battery backup and event logging.

Power Factor

As reactive loads draw current out of phase with the line voltage, there is an associated phase angle. The cosine of this angle provides the power factor.

The power factor will never exceed 1.000. Resistive loads will show a very high power factor, while reactive loads, such as motors, will show lower power factors. Loads with great harmonic content (such as most power supplies) will also indicate a low power factor. Power Factor Correcting (PFC) loads will indicate very high power factors.

Calibration

The calibration process assumes the user has the time and understanding to determine the calibration constants. This process can be greatly simplified. The CS5460 has the capability of determining offsets and gains. By commanding the CS5460 to perform an offset calibration, the offset constants can be found very quickly. The calibration program would indicate the measured value being calibrated and allow the user to adjust the constant, without actually having to know what the constant was. When a satisfactory measurement is achieved, the constant would then be written to the 24C01.

The code presented in this application note almost completely fills the first code page of the PIC16C923. The second code page could be dedicated to a calibration program.

Battery Backup

Some users may wish to have the real time clock continue to run, even during a loss of power. This becomes possible by adding a backup battery to power the PIC16C923 and allow the Timer1 oscillator to operate. This would be the time base for the real time clock. The code to use Timer1 and its oscillator has been included. To extend the life of the battery, it would power only the PIC16C923.

Event Logging

The 24C01 provides 128 bytes of non-volatile EEPROM memory. Currently, only 17 bytes are used for storing calibration data, total energy, and a device serial number. The remaining memory could be used to record power line events, such as black-outs, brown-outs, surges and load peaks. With a real time clock, the times of these events could also be recorded. Recording black-outs and brown-outs would require that the backup battery also power the 24C01.

APPENDIX A: EEPROM DATA MAP

Address	Description
0x00	Device Serial Number
0x01	Voltage Offset MSB
0x02	Voltage Offset
0x03	Voltage Offset LSB
0x04	Current Offset MSB
0x05	Current Offset
0x06	Current Offset LSB
0x07	Voltage Gain MSB
0x08	Voltage Gain
0x09	Voltage Gain LSB
0x0A	Current Gain MSB
0x0B	Current Gain
0x0C	Current Gain LSB
0x0D	Watt-Hour MSB
0x0E	Watt-Hour LSB
0x0F	Pulse Rate Gain MSB
0x10	Pulse Rate Gain
0x11	Pulse Rate Gain LSB

APPENDIX B: SOURCES

The LCD routines came from PICDEM-3™. Adjustments may have been made to the segment and common definitions to account for the use of a different LCD panel than was used in PICDEM-3.

The BIN2BCD routine is loosely based on the B2_BCD_Looped routine in BCD.ASM of application note AN544. This function was originally written for the PIC17CXXX family, but it has been modified for the PIC16CXXX family.

The multiply and divide math routines were copied from application note AN617.

The data sheet for the PIC16C923 can be found at http://www.microchip.com. Search for “DS30444E” or “PIC16C923”.

The data sheet for the 24C01 can be found at http://www.microchip.com. Search for “DS20071J” or “24C01”.

The data sheet for the CS5460 can be found at http://www.crystal.com. Search for “DS279PP3” or “CS5460”.

APPENDIX C: SCHEMATICS

FIGURE C-1: PIC16C923 CONNECTIONS

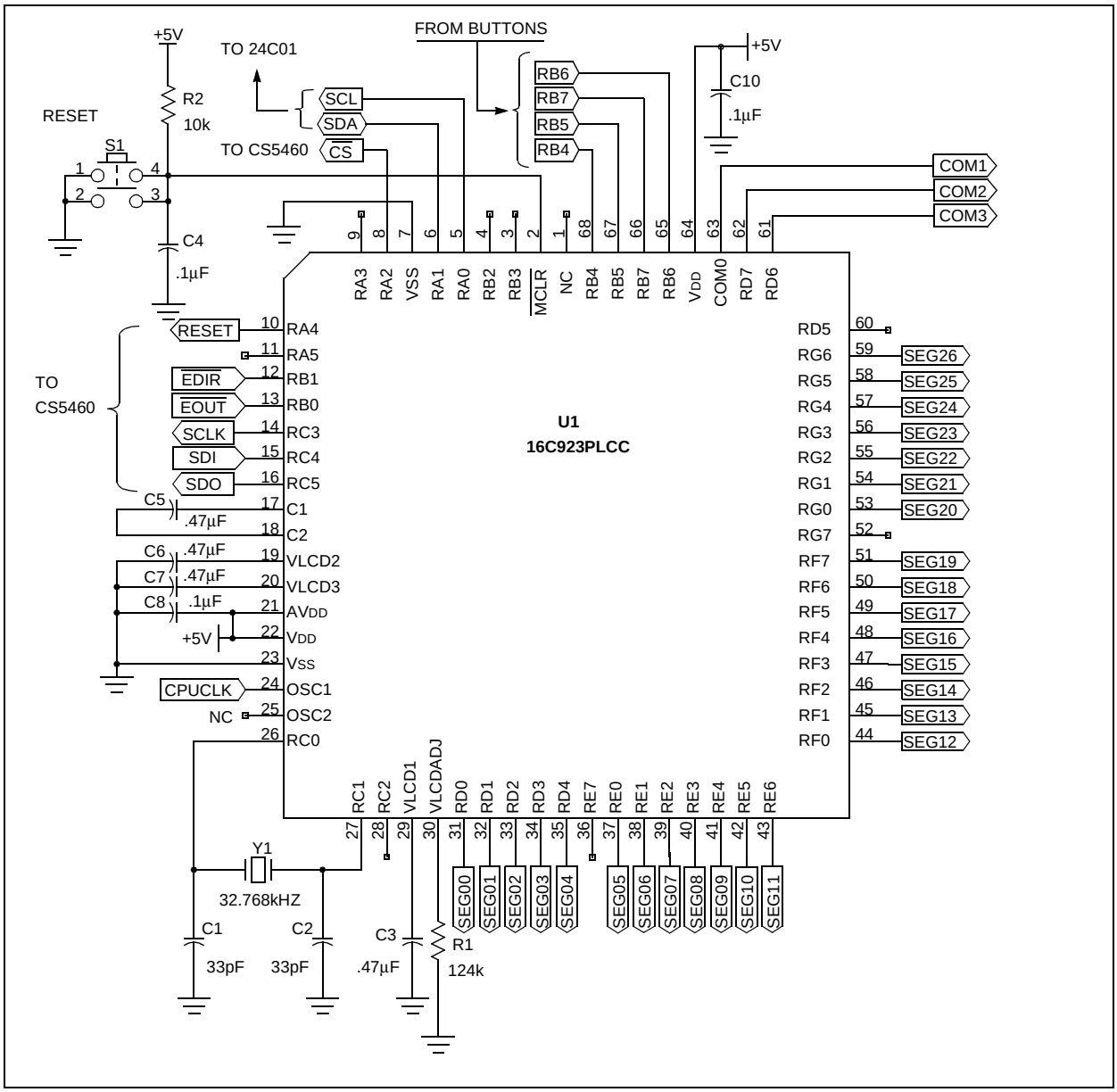


FIGURE C-2: LCD, 24C01, PUSH BUTTONS

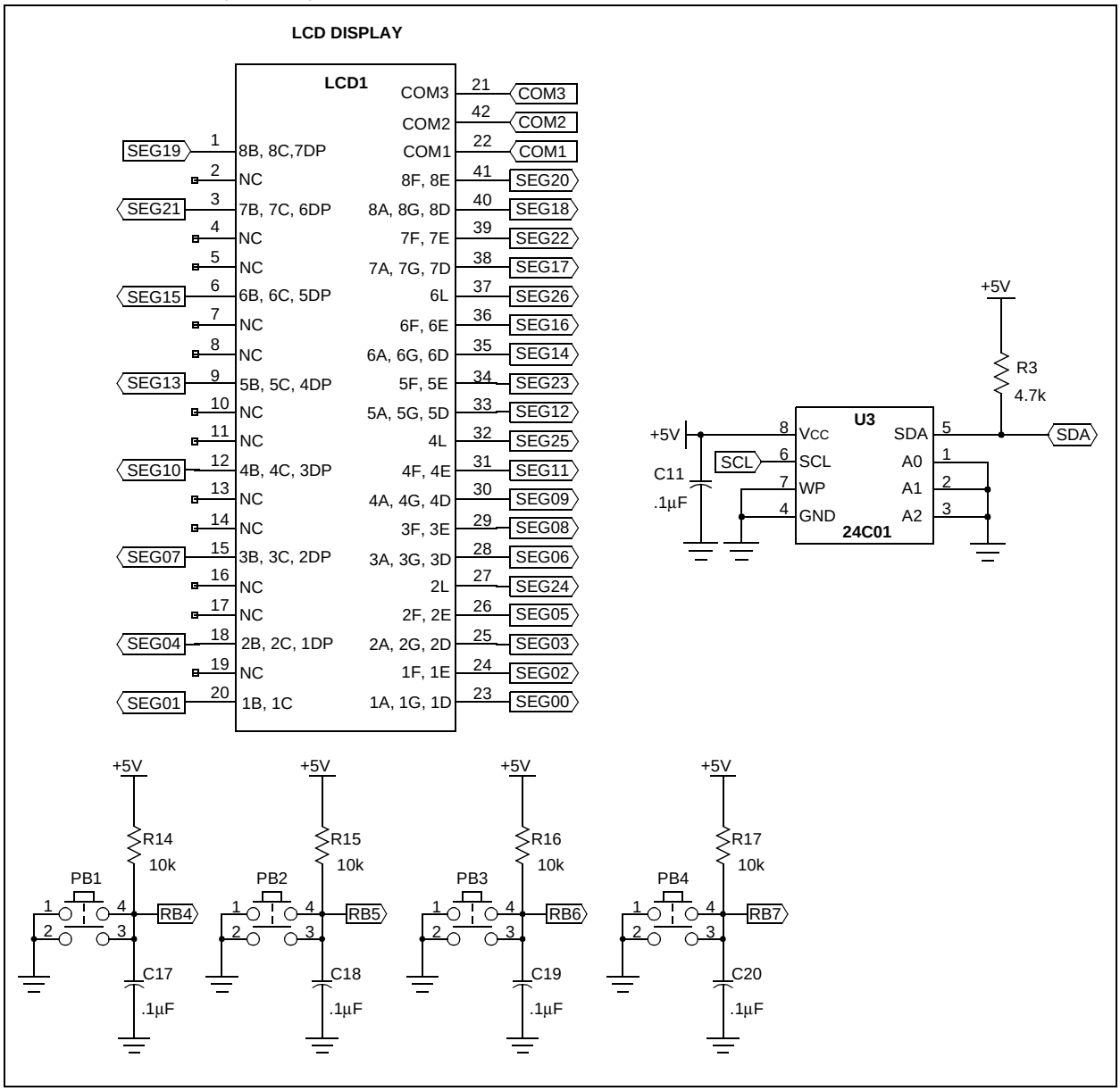


FIGURE C-3: POWER SUPPLY, VOLTAGE SENSE, CURRENT SENSE

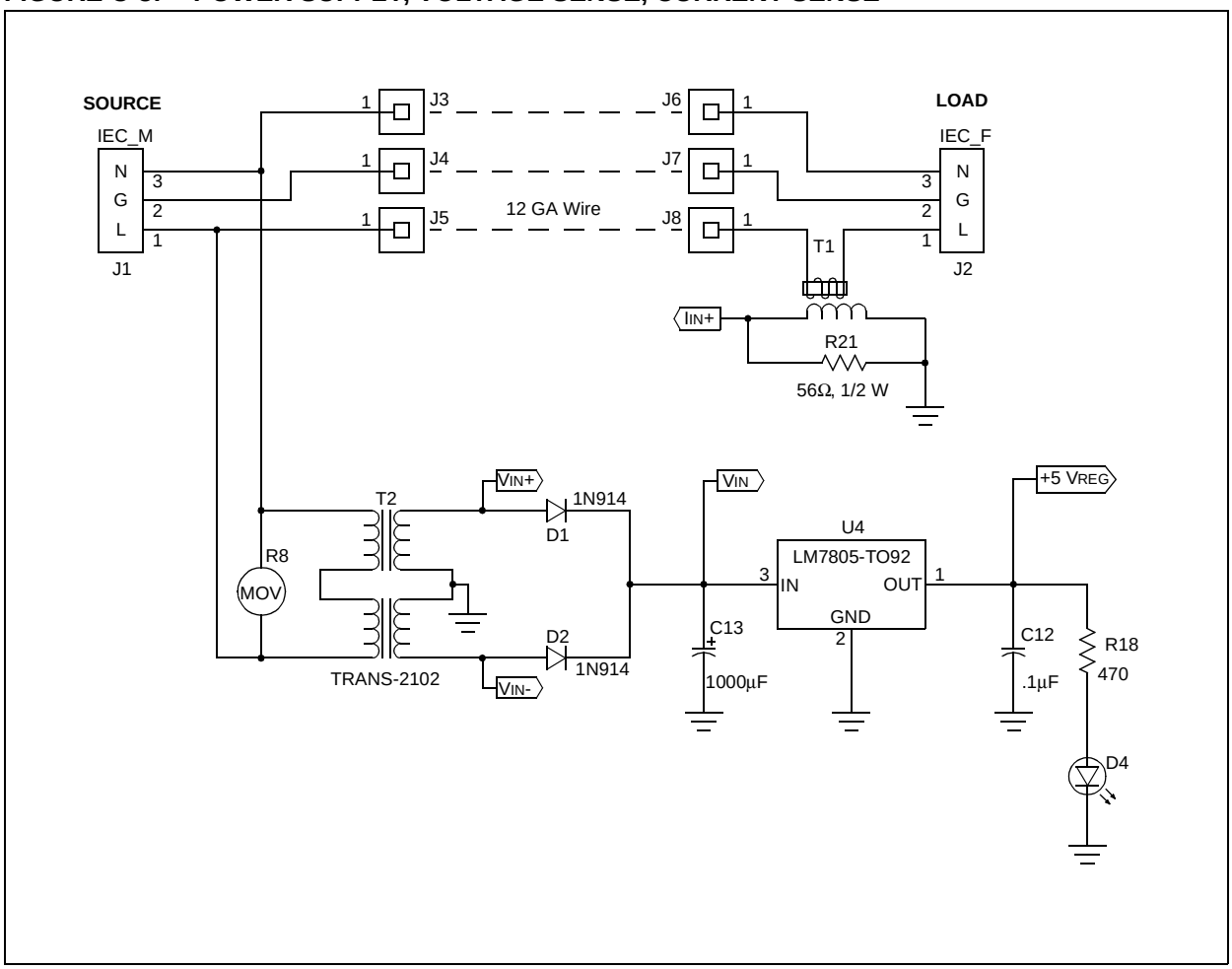
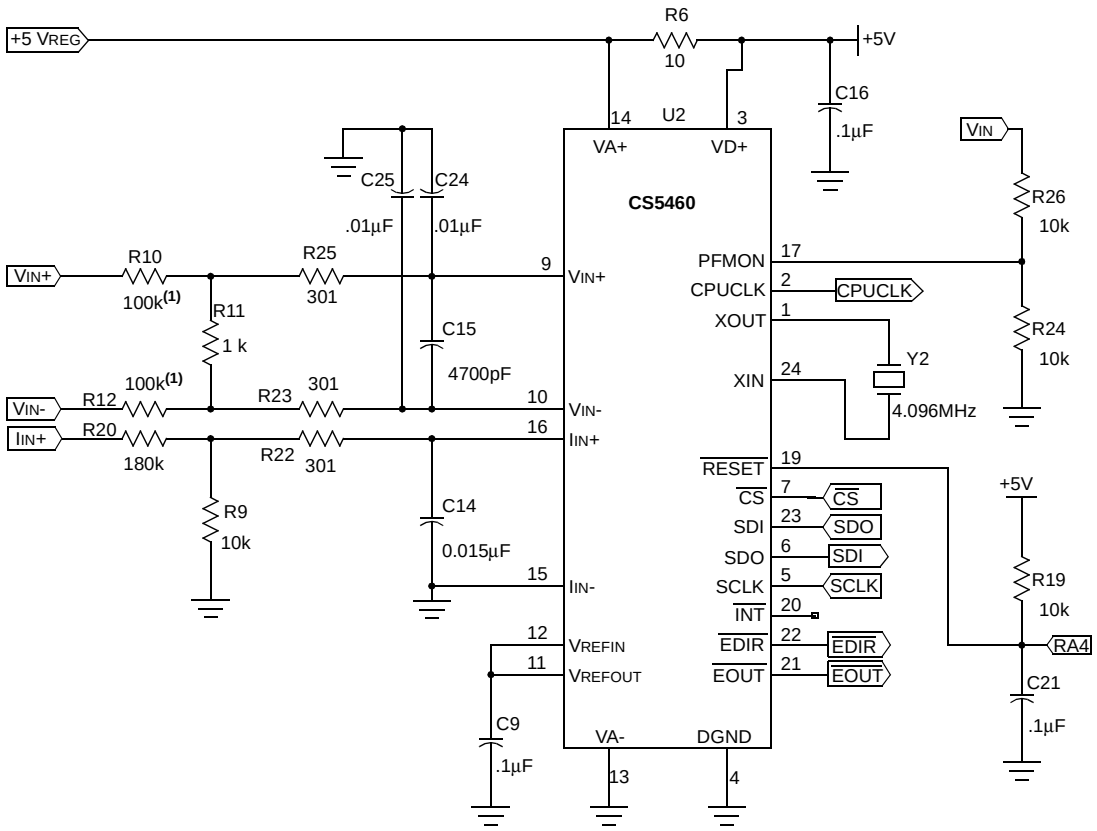


FIGURE C-4: CS5460 CONNECTIONS



Note: R10 and R12 are 121 kOhms for 220V Watt-Hour meters.

APPENDIX D: BILL OF MATERIALS

Cnt	Component Name	RefDes	Description	Digikey
2	1N4001	D1-2	IN4001DICT-ND	
1	16C923PLCC	U1	PIC16C924 (Microchip)	
1	SOCKET	U1	A2144-ND	
1	24C01_TSOP	U3	24CO1-BC (Microchip)	
2	CAP150	C1-2	33pF Capacitor	P4843-ND
12	CAP0805	C4, C8-12, C16-21	.1µF Capacitor	PCC1864CT-ND
2	CAP1206	C24-25	.01µF Capacitor	PCC103BCT-ND
4	CAP1206	C3, C5-7	.47µF Capacitor	PCC1891CT-ND
1	CAP1206	C15	4700pF Capacitor	PCC472BCT-ND
1	CAP1206	C14	15000pF Capacitor	PCC153BCT-ND
1	CAP-RAD400D	C13	1000µF Capacitor	P5142-ND
1	CRYSTAL	Y2	4.096MHz Crystal	X082-ND
1	CRYSTAL_32KHZ	Y1	32kHz Crystal	SE3201-ND
1	CS5460	U2	CS4560 (Crystal Semiconductor)	
1	CSE187-L	T1	Current Transformer	10515-ND
1	IEC_F	J2	Socket	509-1271 (Allied)
1	IEC_M	J1	Plug	509-1269 (Allied)
1	LCD_VIM-808-DP	LCD1	VIM-808-DP-RC-S-HV	153-1057-ND
1	LED_SMT	D4	LED, Green	LT1120CT-ND
1	LM7805-TO92	U4	Voltage Regulator	NJM78L05A-ND
1	MOV	R8	MOV	P7259-ND
1	RES600	R21	56Ω 1/2W Resistor	56H-ND
1	RES1206	R11	1kΩ Resistor	P1.0KECT-ND
1	RES1206	R3	4.7kΩ Resistor	P4.7KECT-ND
1	RES1206	R6	10Ω Resistor	P10ECT-ND
9	RES1206	R2, R9, R14-17, R19, R24, R26	10kΩ Resistor	P10KECT-ND
1	RES1206	R1	124kΩ Resistor	P124KFCT-ND
1	RES1206	R20	121kΩ Resistor	P121KFCT-ND
3	RES1206	R22-23, R25	301Ω Resistor	P300ECT-ND
1	RES1206	R18	470Ω Resistor	P470ECT-ND
2	RES1206	R10, R12	100kΩ Resistor	P100KECT-ND
5	SW-B3F1000	S1, PB1-4		SW404-ND
5	KEY CAP	S1, PB1-4		SW450-ND
1	TRANS-2102	T2	Transformer 12 VAC/0.09 A	MT2113-ND
1	Plastic case			141840 (Jameco)
1	Printed Circuit Board			

Misc:
4 4-40 X 3/8 machine screw for J1, J2
4 4-40 hex nut
2 ft 12 ga stranded copper wire
1 20-pin machined pin IC socket to cut up for pin extensions for S1-S5

APPENDIX E: SOURCE CODE

```
MPASM 02.30.11 Intermediate  WATT_MTR.ASM   5-25-2000  13:31:13         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001               list          st=off         ; suppress list file symbol table
                     00002               list          n=0            ; suppress list file page breaks
                     00003
                     00004 ;*****************************************************************
                     00005 ; Configuration switches. These control what code is assembled.
                     00006
                     00007 ;   Select the desired operating voltage by commenting out all but
                     00008 ;   the desired voltage range
                     00009               #define       VOLT120        ; 120V nominal full scale range
                     00010 ;             #define       VOLT220        ; 220V nominal full scale range
                     00011
                     00012 ;   Selects the real time clock frequency source,
                     00013 ;             #define       TMR1OSC        ; defined if using 32kHz T1OSC, comment out if another RTC source
                     00014
                     00015 ;*********************************************************
                     00016 ; Software License Agreement
                     00017 ;
                     00018 ; The software supplied herewith by Microchip Technology Incorporated (the
                     00019 ; "Company") for its PICmicro® Microcontroller is intended and supplied to
```

```
; you, the Company's customer, for use solely and exclusively on Microchip
; PICmicro Microcontroller products.
;
; The software is owned by the Company and/or its supplier, and is protected
; under applicable copyright laws. All rights are reserved. Any use in
; violation of the foregoing restrictions may subject the user to criminal
; sanctions under applicable laws, as well as to civil liability for the
; breach of the terms and conditions of this license.
;
; THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER
; EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED
; WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY
; TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE
; FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.


;************************************************************
; Author: Stephen Humberd, Brett Duane
; Company: Microchip Technology Inc.
; Revision: 1.1
; Date: 5-15-2000
; Assembled using MPLAB 4.99.07, MPASM 2.30.11
;************************************************************
; Include Files: CAL.INC      calibration constants
;                PR3.INC      LCD segment definitions
;                P16C924.INC  Standard Microchip include file for PIC16C923/924
;************************************************************
; This program controls and reads data from The Crystal CS4560
; Single Phase Bi-Directional Power/Energy IC
; and displays it on a eight digit LCD using the LCD drive
; function of a PIC16C923.

; The CS5460 measures line voltage and current transfered between
; the line (source) and the load. The instantanious voltage and current
; measurements are used to compute (within the CS5460) instantanious power,
; RMS voltage, RMS current, and accumulated energy. All of these
; measurements and calculation results are avalible to the PIC16C923 via SPI.
;
; The RMS voltage and RMS current are displayed and used to calculate aparent
; power.
;
; A pulse output (EOUT) on the CS5460 indicates when a programable amout of
; energy has been transferred between the line and the load. Another output
; (EDIR) indicates the direction of that transfer (if a load is highly
; reactive, enegry flows from the load to the line). These pulses are counted
; by the PIC16C923 to measure and display total energy transferred in WattHours
```

```
00066 ;
00067 ; True power is measured by counting pulses for 1 second. The CS5460 has been
00068 ; programmed to generate a pulse for each 10WattSeconds of energy tranferred.
00069 ; The pulse count is multiplied by 10 to calculate true power.
00070 ;
00071 ; Aparent power and true power are displayed, and are used to calculate the
00072 ; power factor of the load.
00073 ;
00074 ; The CS4560 outputs a 4.096MHz clock for use by the PIC16C923 as the system
00075 ; clock. This code offers the option of using this source as the real time clock
00076 ; source with CCP1 in compare/interrupt only/special event mode, or using the Timer1
00077 ; oscillator with a 32.768KHz crystal. The hardware provided on the demo units
00078 ; supports both options.
00079 ;
00080 ; Every 8 minutes, the current accumulated energy (WHr) is written to the 24C01
00081 ; serial EEPROM. This saves the total energy during times when the AC power is removed.
00082 ;
00083 ; Calibration constants are also stored in the 24C01. These are read from the 24C01
00084 ; and written back to the CS5460 when power is reapplied, and when the total enegy is
00085 ; cleared from the 24C01. The code to read the constants from cal.inc and write them to
00086 ; the 24C01 has been included, but has been commented out.
00087 ;
00088 ; On reset, the PIC16C923 checks the 4 control buttons for 3 specific states. One state
00089 ; allows the user to set the time of the real time clock. Another state clears the
00090 ; total WHr for the 24C01 and rewrites the calibration constants to the CS5460. A
00091 ; third state allows the user to adjust the calibration constants in the CS5460.
00092 ;
00093 ; Written by Stephen Humberd, Microchip Technology 10/08/1999
00094 ;*************************************************************
00095 ; Optional Real Time Clock sources (T1OSC or CS5460 CPUCLK output)
00096 ; Monitor CS5460 !EDIR output.
00097 ; Change Pulse Rate from 128 pulses/KWHr (1 pulse/28,125Wsec)
00098 ;    to 1 pulse/10Wsec (100W load generates 10 pulses per second)
00099 ; Use 16-bits of CS5460 data rather than 24-bits
00100 ;    (CS5460 settings are still 24-bits long)
00101 ; Added Apparent Power, True Power, and Power Factor Functions.
00102 ; Moved Pulse Rate register value to EEPROM.
00103 ; Modified Calibrate routine to include Pulse Rate "Gain" function.
00104 ; General code size reduction. (Fits in PIC16C923.)
00105 ;
00106 ; If the CPUCLK output of the CS5460 is 4.096MHz, the CCP1 module
00107 ; can be used (as is) for the real time clock reference instead of the
00108 ; Timer1 Oscillator. If the CPUCLK output is some other frequency,
00109 ; CCPR1H:CCPR1L will need to be adjusted to make CCP1 generate an interrupt
00110 ; every 0.5 second.
00111 ;
```

```
                    00112 ; If the Timer1 oscillator is to be used as the Real Time Clock base,
                    00113 ; uncomment line 46 ("#define TMR1OSC"). This causes CCP1 code to
                    00114 ; be disabled and T1OSC code to be enabled.
                    00115 ;
                    00116 ; Modified by Brett Duane, Microchip Technology 5/25/2000
                    00117 ;*****************************************************************
                    00118
                    00119                 list p=16c924
                    00120                 #include <p16c924.inc>
                    00001         LIST
                    00002 ; P16C924.INC  Standard Header File, Version 1.01    Microchip Technology, Inc.
                    00289         LIST
2007   3FF1         00121                 __CONFIG _CP_OFF&_WDT_OFF&_XT_OSC&_PWRTE_ON
                    00122
                    00123                 errorlevel     -302           ; suppress assembler warning message "Operand not in bank 0"
                    00124                 errorlevel     -306           ; suppress assembler warning message "Crossing page boundary"
                    00125
                    00126
                    00127 ;*****************************************************************
                    00128 ; various equates
                    00129
                    00130 ; Microchip MATH library AN617
00000007            00131 MSB             equ            7
00000000            00132 LSB             equ            0
                    00133
                    00134 ; CS5460 variables (string equates)
000000FE            00135 SYNC0           equ            0xFE           ; Sync bytes sent as dummy data during reads
000000FF            00136 SYNC1           equ            0xFF           ; No more than 3 SYNC1 bytes at a time.
                    00137
                    00138                 #IFNDEF        TMR1OSC
0000007F            00139 CCPCOUNT        equ            0x7F           ; Variable in shared RAM. CCP1 interrupt counter,
                    00140                 ENDIF                         ; not used with Timer1 oscillator.
                    00141
                    00142 ;*****************************************************************
                    00143 ; Variables in RAM
                    00144                 cblock         0x20           ; Load variables into RAM starting at 0x20
00000020            00145 TEMPA           ; scratchpad
00000021            00146 TEMPB           ; scratchpad
                    00147
00000022            00148 SECOND          ; TIME variables
00000023            00149 MINUTE          ; TIME variables
00000024            00150 HOUR            ; TIME variables
                    00151
00000025            00152 WATTTMPH        ; energy pulse counter MSB
00000026            00153 WATTTMPL        ; energy pulse counter LSB
00000027            00154 WATTHRH         ; Accumulated WHr MSB
```

```
00000028            00155 WATTHRL        ; Accumulated WHr LSB
                    00156
00000029            00157 PULSECH        ; "Pulses per second" counter MSB
0000002A            00158 PULSECL        ; "Pulses per second" counter LSB
                    00159
0000002B            00160 PULDISPH       ; "Pulses during previous second" MSB
0000002C            00161 PULDISPL       ; "Pulses during previous second" LSB
                    00162
0000002D            00163 APH            ; Apparent Power MSB (16 bit) (VoltAmps in binary)
0000002E            00164 APL            ; Apparent Power LSB
                    00165
0000002F            00166 TPH            ; True Power MSB (16 bit) (Watts in binary)
00000030            00167 TPL            ; True Power LSB
                    00168
00000031            00169 PFH            ; Power Factor MSB (16 bit) (unitless in binary)
00000032            00170 PFL            ; Power Factor LSB
                    00171
                    00172 ;More general variables
00000033            00173 POINTER        ; Points to the characters in initial
00000034            00174 PTRTMP         ; scrolling message
00000035            00175 SERNUM         ; unit serial number
00000036            00176 CALMODE        ; Calibration Mode being displayed
00000037            00177 CALDIG         ; Calibration digit being incremented
00000038            00178 UPDATEWH       ; When to write accumulated Watt HOURs to the EEPROM (every 8 minutes)
                    00179
                    00180 ; SSP variables (SPI mode) Receive buffer from CS5460
00000039            00181 RXDATA0        ; RXDATA buffer MSB
0000003A            00182 RXDATA1        ; RXDATA buffer
0000003B            00183 RXDATA2        ; RXDATA buffer LSB
                    00184
                    00185 ; TXDATA buffer (SPI mode) Transmit buffer to CS5460
0000003C            00186 TXDATA         ; command byte to send to the CS4560
0000003D            00187 TXDATA0        ; TXDATA buffer MSB
0000003E            00188 TXDATA1        ; TXDATA buffer
0000003F            00189 TXDATA2        ; TXDATA buffer LSB
                    00190
                    00191 ; Microchip MATH library AN617
00000040            00192 AARGB0         ; 4 byte argument and result. B0 is always MSB
00000041            00193 AARGB1
00000042            00194 AARGB2
00000043            00195 AARGB3
00000044            00196 AARGB4
                    00197
00000045            00198 BARGB0         ; 2 byte argument. B0 is always MSB
00000046            00199 BARGB1
                    00200
```

```
00000047     REMB0          ; 2 byte remaider (from division)
00000048     REMB1          ; B0 is always MSB

00000049     TEMP           ; internal variables
0000004A     TEMPB0
0000004B     TEMPB1
0000004C     TEMPB2
0000004D     LOOPCOUNT

             ; 16 bit binary to BCD variables - See apnote AN544
0000004E     R0          ; output bytes 10k’s digits
0000004F     R1          ;                1k’s, 100’s digits
00000050     R2          ;                10’s,   1’s digits

00000051     TEMPL          ; input bytes (2 byte binary)
00000052     TEMPH

00000053     COUNT          ; internal variable

00000054     BYTECOUNT      ; EEPROM variables (firmware I2C)
00000055     EEADDR         ; EEPROM addresses
00000056     EEDATA         ; EEPROM data
00000057     EETEMP

             ;powerup variables
00000058     PWRUP55        ; registers to test at power up to
00000059     PWRUPAA        ; see if warm start or cold start

0000005A     LASTRAM        ; dummy marker (unused)

                            endc

                            cblock         0x70           ; Load variables into RAM starting at 0x70
             ; "SHARED" variables in memory locations 70h-7Fh are avaliable in all banks

00000070     LCDTEMP1       ; used by LCD routines
00000071     LCDTEMP2

00000072     MODEINC        ; incremented each second,
                            ; bit 7 set whever Bit 0 is set (every odd second)

00000073     UPDATE         ; Bit 0 set each second

00000074     BUTTON         ; indicates button press. Bit0=1 = a button is pressed
00000075     BUTTONTMP      ; used only in ISR.       Bits<1:4>=1 = buttons 1-4 are pressed
```

```
00000076                00247 ; Save context before interrupt, restore after interrupt
00000077                00248 TEMPFSR        ; save FSR register during ISR
00000078                00249 TEMPPCLATH     ; save PCLATH register during ISR
00000079                00250 TEMPW          ; save W register during ISR
                        00251 TEMPSTAT       ; save STATUS register during ISR
                        00252                endc
                        00253
                        00254                #include       <cal.inc>      ; calibration definitions
                        00001 ; Software License Agreement
                        00002 ;
                        00003 ; The software supplied herewith by Microchip Technology Incorporated (the
                        00004 ; "Company") for its PICmicro® Microcontroller is intended and supplied to
                        00005 ; you, the Company's customer, for use solely and exclusively on Microchip
                        00006 ; PICmicro Microcontroller products.
                        00007 ;
                        00008 ; The software is owned by the Company and/or its supplier, and is protected
                        00009 ; under applicable copyright laws. All rights are reserved. Any use in
                        00010 ; violation of the foregoing restrictions may subject the user to criminal
                        00011 ; sanctions under applicable laws, as well as to civil liability for the
                        00012 ; breach of the terms and conditions of this license.
                        00013 ;
                        00014 ; THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER
                        00015 ; EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED
                        00016 ; WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY
                        00017 ; TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE
                        00018 ; FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
                        00019
                        00020 ; ************  DEFAULT VALUES TO WRITE TO EEPROM  *************************
                        00021 ; After the calibration values have been determined for a new
                        00022 ; unit those values can be loaded into these "#defines" and
                        00023 ; written to the EEPROM (see Calibration instruction document)
                        00024
                        00025 #define        SERNUMBER      0x01           ; Serial Number
                        00026
                        00027 #define        VOLTGAINH      0x29           ; voltage gain MSB
                        00028 #define        VOLTGAINM      0x66           ; voltage gain
                        00029 #define        VOLTGAINL      0x6F           ; voltage gain LSB
                        00030
                        00031 #define        CURRGAINH      0x2A           ; current gain MSB
                        00032 #define        CURRGAINM      0x2E           ; current gain
                        00033 #define        CURRGAINL      0x0A           ; current gain LSB
                        00034
                        00035 #define        VOLTOFFH       0x10           ; voltage offset MSB
                        00036 #define        VOLTOFFM       0x1A           ; voltage offset
                        00037 #define        VOLTOFFL       0xB6           ; voltage offset LSB
                        00038
```

```
#define        CURROFFH       0xFE           ; current offset MSB
#define        CURROFFM       0xDC           ; current offset
#define        CURROFFL       0x16           ; current offset LSB
#define        PULSERATEH     0x02           ; Pulse rate "gain" MSB
#define        PULSERATEM     0x14           ; Pulse rate "gain"
#define        PULSERATEL     0xCB           ; Pulse rate "gain" LSB
; *************** FULL SCALE VALUES *************************************
; The present rev of the CS4560 is not linear over the entire voltage range
; Values used by the MATH routines for coarse calibration
; based on the values we choose for the voltage divider
#define        MAXCURRH       0x00           ; Full scale current MSB ... 25.0 Amps
#define        MAXCURRL       0xFA           ; Full scale current LSB
#IFDEF         VOLT120        ; 120V nominal full scale
; for 120V R10 & R12 = 100K, R11 = 1K
; use this value for full scale voltage
#define        MAXVOLTH       0x00           ; Full scale voltage MSB ... 237 VOLTs
#define        MAXVOLTL       0xED           ; Full scale voltage LSB
#define        MAXPWRH        0x98           ; Full scale power MSB ... 237V * 25.0A = 5925 W
#define        MAXPWRL        0xCE           ; Full scale power LSB
#ENDIF
#IFDEF         VOLT220        ; 220V nominal full scale
; for 220V R10 & R12 = 121K, R11 = 1K
; use this value for full scale voltage
#define        MAXVOLTH       0x01           ; Full scale voltage MSB ... 282 VOLTs
#define        MAXVOLTL       0x1A           ; Full scale voltage LSB
#define        MAXPWRH        0xB3           ; Full scale power MSB ... 282V * 25.0A = 7050 W
#define        MAXPWRL        0xF1           ; Full scale power LSB
#ENDIF
               #include       <pr3a.inc>     ; LCD, I2C, and math library definitions
;*************************************************************
; Software License Agreement
;
; The software supplied herewith by Microchip Technology Incorporated (the
; "Company") for its PICmicro® Microcontroller is intended and supplied to
```

```
00006 ; you, the Company's customer, for use solely and exclusively on Microchip
00007 ; PICmicro Microcontroller products.
00008 ;
00009 ; The software is owned by the Company and/or its supplier, and is protected
00010 ; under applicable copyright laws. All rights are reserved. Any use in
00011 ; violation of the foregoing restrictions may subject the user to criminal
00012 ; sanctions under applicable laws, as well as to civil liability for the
00013 ; breach of the terms and conditions of this license.
00014 ;
00015 ; THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER
00016 ; EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED
00017 ; WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY
00018 ; TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE
00019 ; FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
00020
00021 ;***************** EEPROM I2C communications ****************************
00022 #define        SCL             PORTA,0
00023 #define        SDA             PORTA,1
00024 #define        SDATRIS         TRISA,1
00025
00026 ;***************** SSP **************************************************
00027 #define        CS              PORTA,2
00028
00029 ;***************** LCD **************************************************
00030 ; We define what bit in what register corresponds to each LCD segment
00031
00032 #define        D1A             LCDD02,2        ; DIGIT 1 (LEFT SIDE)
00033 #define        D1B             LCDD02,3
00034 #define        D1C             LCDD06,3
00035 #define        D1D             LCDD10,2
00036 #define        D1E             LCDD06,4
00037 #define        D1F             LCDD02,4
00038 #define        D1G             LCDD06,2
00039
00040 #define        D2A             LCDD02,1        ; DIGIT 2
00041 #define        D2B             LCDD02,5
00042 #define        D2C             LCDD06,5
00043 #define        D2D             LCDD10,1
00044 #define        D2E             LCDD06,6
00045 #define        D2F             LCDD02,6
00046 #define        D2G             LCDD06,1
00047
00048 #define        D3A             LCDD01,6        ; DIGIT 3
00049 #define        D3B             LCDD01,7
00050 #define        D3C             LCDD05,7
00051 #define        D3D             LCDD09,6
```

```
#define        D3E           LCDD06,0
#define        D3F           LCDD02,0
#define        D3G           LCDD05,6

#define        D4A           LCDD01,4        ; DIGIT 4
#define        D4B           LCDD01,5
#define        D4C           LCDD05,5
#define        D4D           LCDD09,4
#define        D4E           LCDD06,7
#define        D4F           LCDD02,7
#define        D4G           LCDD05,4

#define        D5A           LCDD01,1        ; DIGIT 5
#define        D5B           LCDD01,2
#define        D5C           LCDD05,2
#define        D5D           LCDD09,1
#define        D5E           LCDD05,3
#define        D5F           LCDD01,3
#define        D5G           LCDD05,1

#define        D6A           LCDD00,6        ; DIGIT 6
#define        D6B           LCDD00,7
#define        D6C           LCDD04,7
#define        D6D           LCDD08,6
#define        D6E           LCDD05,0
#define        D6F           LCDD01,0
#define        D6G           LCDD04,6

#define        D7A           LCDD00,3        ; DIGIT 7
#define        D7B           LCDD00,4
#define        D7C           LCDD04,4
#define        D7D           LCDD08,3
#define        D7E           LCDD04,5
#define        D7F           LCDD00,5
#define        D7G           LCDD04,3

#define        D8A           LCDD00,0        ; DIGIT 8 (RIGHT SIDE)
#define        D8B           LCDD00,1
#define        D8C           LCDD04,1
#define        D8D           LCDD08,0
#define        D8E           LCDD04,2
#define        D8F           LCDD00,2
#define        D8G           LCDD04,0

#define        COLON1        LCDD11,2        ; COLONS
#define        COLON2        LCDD11,1
```

```
                    00098 #define        COLON3          LCDD11,0
                    00099
                    00100 #define        DEC1            LCDD10,3        ; DECIMAL POINTS
                    00101 #define        DEC2            LCDD10,5
                    00102 #define        DEC3            LCDD09,7
                    00103 #define        DEC4            LCDD09,5
                    00104 #define        DEC5            LCDD09,2
                    00105 #define        DEC6            LCDD08,7
                    00106 #define        DEC7            LCDD08,4
                    00107
                    00108
                    00256
                    00257 ;*************************************************************************
                    00258 ; Reset and interrupt vectors                                            *
                    00259 ;*************************************************************************
                    00260
0000                00261                org             0x0000          ; Reset vector
0000   28B2         00262                goto            Start
                    00263
0004                00264                org             0x0004          ; Interrupt vector
                    00265
0004   00F8         00266                movwf           TEMPW           ; save context before executing
0005   0E03         00267                swapf           STATUS,W        ; ISR (Interrupt Service Routine)
0006   00F9         00268                movwf           TEMPSTAT
0007   0804         00269                movf            FSR,W
0008   00F6         00270                movwf           TEMPFSR
0009   080A         00271                movf            PCLATH,W
000A   00F7         00272                movwf           TEMPPCLATH
                    00273
000B   018A         00274                clrf            PCLATH          ; Prog page 0
000C   1283         00275                bcf             STATUS,RP0      ; SFR bank 0
000D   1303         00276                bcf             STATUS,RP1      ; SFR bank 0
                    00277
                    00278 ; Find out which flag caused the interrupt, then branch to apropriate subroutine
                    00279
000E   188B         00280                btfsc           INTCON,INTF     ; RB0/INT flag
000F   286F         00281                goto            RB0ISR          ; Pulse from CS4560 EOUT pin
                    00282
                    00283 #IFDEF TMR1OSC
                    00284                btfsc           PIR1,TMR1IF     ; Check Timer1 interrupt flag
                    00285 #ELSE
0010   190C         00286                btfsc           PIR1,CCP1IF     ; Check CCP1 interrupt flag, not used with Timer1
                    00287 #ENDIF
0011   2818         00288                goto            TMR1ISR         ; Timer1 overflowed (One second timer)
                    00289
0012   180B         00290                btfsc           INTCON,RBIF     ; Check PORTB flag
```

```
0013   2847            00291                goto           ButtonISR      ; A BUTTON was pressed
                       00292
0014   188C            00293                btfsc          PIR1,TMR2IF    ; Check Timer2 flag
0015   2853            00294                goto           TMR2ISR        ; debounce time for BUTTONs
                       00295
                       00296                ; if you get here it wasn't a real interrupt, go back to the main program
0016   128B            00297                bcf            INTCON,T0IE    ; disable timer0 interrupts
                       00298
0017   28A9            00299                goto           POP            ; restore context, return from interrupt
                       00300
                       00301 ;*****************************************************************
                       00302 ; Timer1 overflow interrupt
                       00303 ; This is the interrupt for the second counter for the Real Time Clock
                       00304 ; If T1OSC used, the timer is stopped, a new value is reloaded, and the timer
                       00305 ; is restarted.
                       00306 ;
                       00307 ; If CCP1 is used, only an interrupt is generated, but interrupts occur twice
                       00308 ; as often as needed. The interrupt rate is dived by 2 to get a 1 second interrupt
                       00309 ; rate. No reloading is required.
                       00310 ;
                       00311 ; This interrupt is the basis of the 1 second timer tick that regulates the various
                       00312 ; display modules, and the real time clock.
                       00313 ;
                       00314 ; The count in the PULSEC<H:L> variables is moved to PULDISP<H:L>, and PULSEC<H:L> is cleared.
                       00315 ;
                       00316 ; Input variables:  CCPCOUNT, MODEINC, SECOND, MINUTE, HOUR, WATTHRL, WATTHRH
                       00317 ; Output variables: UPDATE, PULSECL, PULSECH, WATTHRL, WATTHRH, UPDATEWH
                       00318
0018                   00319 TMR1ISR
                       00320 #IFDEF TMR1OSC ; use this code if 32kHz Timer1OSC is used
                       00321
                       00322                bcf            T1CON,TMR1ON   ; turn off timer
                       00323                movlw          0x7f                          ; reload Timer1 registers
                       00324                movwf          TMR1H
                       00325                movlw          0xff
                       00326                movwf          TMR1L
                       00327                bsf            T1CON,TMR1ON   ; turn on timer
                       00328
                       00329                bcf            PIR1, TMR1IF   ; Clear TMR1 interrupt flag
                       00330
                       00331 #ELSE                         ; use this code if the system clock is used
                       00332
0018   110C            00333                bcf            PIR1, CCP1IF   ; Clear CCP1 interrupt flag
                       00334
0019   0AFF            00335                incf           CCPCOUNT,F                    ; count interrupt
001A   187F            00336                btfsc          CCPCOUNT,0                    ; divide interrupt rate by 2
```

```
001B   28A9            00337                goto            POP                         ; not yet time for 1 second interrupt (1/2 second)
                       00338
                       00339 #ENDIF
                       00340
001C   1473            00341                bsf             UPDATE,0                    ; Set to update display
001D   0AF2            00342                incf            MODEINC,F
                       00343
                       00344 ; This is how often we change display MODEs
                       00345 ; (time, volts, current, etc)
                       00346
001E   18F2            00347                btfsc           MODEINC,1                   ; set to change MODE every 2 seconds
001F   17F2            00348                bsf             MODEINC,7
                       00349
0020   082A            00350                movf            PULSECL,W                   ; COUNT the pulses each second from the
0021   00AC            00351                movwf           PULDISPL                    ; CS4560 EOUT pin.  Every second move
0022   0829            00352                movf            PULSECH,W                   ; the count from the counter variable (PULSEC#)
0023   00AB            00353                movwf           PULDISPH                    ; to the display.  (PULDISP#)
0024   01AA            00354                clrf            PULSECL                     ; clear counter (PULSEC#)
0025   01A9            00355                clrf            PULSECH
                       00356
0026   0AA2            00357                incf            SECOND,F                    ; increment seconds
0027   0822            00358                movf            SECOND,W
0028   3C3C            00359                sublw           .60                         ; see if we have counted 60 seconds
0029   1903            00360                btfsc           STATUS,Z
002A   282C            00361                goto            IncMINUTE                   ; if we have then increment minutes
002B   28A9            00362                goto            POP
002C                   00363 IncMINUTE
002C   01A2            00364                clrf            SECOND                      ; clear SECONDs counter every minute
002D   0AB8            00365                incf            UPDATEWH,F
002E   1DB8            00366                btfss           UPDATEWH,3                  ; Write Watt HOURs to EEPROM every 8 minutes
002F   283A            00367                goto            IncMINUTE2                  ; (bit3 of the update Watt HOUR counter)
0030   0828            00368                movf            WATTHRL,W                   ; Accumulated Watt HOURs LSB
0031   00D6            00369                movwf           EEDATA                      ; put the data in EEDATA
0032   300D            00370                movlw           0x0D                        ; the EEPROM address in EEADDR
0033   00D5            00371                movwf           EEADDR
0034   241A            00372                call            EEWrite                     ; call EEWrite to write it
0035   0827            00373                movf            WATTHRH,W                   ; MSB
0036   00D6            00374                movwf           EEDATA
0037   0AD5            00375                incf            EEADDR,F
0038   241A            00376                call            EEWrite
0039   01B8            00377                clrf            UPDATEWH
003A                   00378 IncMINUTE2
003A   0AA3            00379                incf            MINUTE,F        ; increment MINUTEs
003B   0823            00380                movf            MINUTE,W
003C   3C3C            00381                sublw           .60             ; if we have counted 60 minutes
003D   1903            00382                btfsc           STATUS,Z
```

```
003E   2840           00383                goto           IncHOUR        ; then increment the HOURs
003F   28A9           00384                goto           POP
0040                  00385 IncHOUR
0040   01A3           00386                clrf           MINUTE
0041   0AA4           00387                incf           HOUR,F         ; increment HOURs
0042   0824           00388                movf           HOUR,W
0043   3C18           00389                sublw          .24            ; if we have counted 24 hours then
0044   1903           00390                btfsc          STATUS,Z
0045   01A4           00391                clrf           HOUR           ; reset HOUR to zero
0046   28A9           00392                goto           POP
                      00393
                      00394 ;*****************************************************************************
0047                  00395 ButtonISR      ; ISR - A button was pressed, save BUTTON state in BUTTONTMP, start Timer2 for debounce
                      00396
0047   0806           00397                movf           PORTB,W        ; Read the Port
0048   39F0           00398                andlw          0xf0           ; clear lower nibble (BUTTONs are on RB4-7)
0049   00F5           00399                movwf          BUTTONTMP      ; save nibble
                      00400
004A   100B           00401                bcf            INTCON,RBIF    ; clear flag (already read PORTB)
                      00402
004B   118B           00403                bcf            INTCON,RBIE    ; disable PORTB interrupt
                      00404
004C   0191           00405                clrf           TMR2           ; clear Timer2 count
004D   108C           00406                bcf            PIR1,TMR2IF    ; clear timer2 interrupt flag
                      00407
004E   1683           00408                bsf            STATUS,RP0     ; Bank 1
004F   148C           00409                bsf            PIE1,TMR2IE    ; turn on timer2 interrupt (debounce time)
0050   1283           00410                bcf            STATUS,RP0     ; Bank 0
                      00411
0051   1512           00412                bsf            T2CON,TMR2ON   ; start timer2
                      00413
0052   28A9           00414                goto           POP
                      00415
                      00416 ;*****************************************************************************
                      00417 ;Debounce time for BUTTONs
                      00418 ; After Timer2 timeout read the BUTTONs (RB4-7) again
                      00419 ;
                      00420 ; When a button is first detected, PORTB interrupt on change is disabled, the
                      00421 ; button state is saved, and Timer2 is started.
                      00422 ;
                      00423 ; When Timer2 overflows, this ISR disables the timer, and checks to see if the
                      00424 ; button state has changed.
                      00425 ;
                      00426 ; If no changes are detected, the button press is written to BUTTON, otherwise
                      00427 ; the button press is discarded.
                      00428 ;
```

```
                        00429 ; PORTB interrupt on change is then re-enabled.
                        00430 ;***********************************************************************
                        00431 ; Inputs:  PORTB, BUTTONTMP
                        00432 ; Outputs: BUTTON
                        00433
0053                    00434 TMR2ISR
0053   1112             00435                 bcf             T2CON,TMR2ON    ; turn off timer2
                        00436
0054   1683             00437                 bsf             STATUS,RP0                      ; bank 1
0055   108C             00438                 bcf             PIE1,TMR2IE                     ; disable timer2 interrupt
0056   1283             00439                 bcf             STATUS,RP0                      ; bank 0
                        00440
0057   108C             00441                 bcf             PIR1,TMR2IF                     ; clear timer2 interrupt flag
                        00442
0058   01F4             00443                 clrf            BUTTON                          ; clear button status
                        00444
0059   0806             00445                 movf            PORTB,W                         ; read PORTB
005A   100B             00446                 bcf             INTCON,RBIF                     ; clear PORTB interrupt flag
005B   158B             00447                 bsf             INTCON,RBIE                     ; enable PORTB interrupt on change
                        00448
005C   39F0             00449                 andlw           0xf0                            ; clear lower nibble
005D   0275             00450                 subwf           BUTTONTMP,W                     ; compare with last button state
005E   1903             00451                 btfsc           STATUS,Z                        ; did previous and present button states match?
005F   2862             00452                 goto            ChkButtons                      ; debounce good ... read buttons
                        00453
0060   01F5             00454                 clrf            BUTTONTMP                       ; debounce failed ... ignore
0061   28A9             00455                 goto            POP
                        00456
                        00457 ; The variable "BUTTON" returns the BUTTON press status
                        00458 ; bit0   = at least one BUTTON was pressed
                        00459 ; bit1-4 = BUTTON(s) that was pressed (bit1 set = BUTTON 1 pressed etc)
                        00460
0062                    00461 ChkButtons
0062   0875             00462                 movf            BUTTONTMP,W                     ; test previous state for any pressed buttons
0063   1903             00463                 btfsc           STATUS,Z                        ; ignore button release
0064   28A9             00464                 goto            POP
                        00465
0065   1474             00466                 bsf             BUTTON,0                        ; a button was pressed
0066                    00467 Button1
0066   1E75             00468                 btfss           BUTTONTMP,4                     ; test button 1
0067   14F4             00469                 bsf             BUTTON,1                        ; button 1 was pressed
0068                    00470 Button2
0068   1EF5             00471                 btfss           BUTTONTMP,5                     ; test button 2
0069   1574             00472                 bsf             BUTTON,2                        ; button2 was pressed
006A                    00473 Button3
006A   1F75             00474                 btfss           BUTTONTMP,6                     ; test button 3
```

```
006B   15F4          00475                bsf            BUTTON,3                      ; button 3 was pressed
006C                 00476 Button4
006C   1FF5          00477                btfss          BUTTONTMP,7                   ; test button 4
006D   1674          00478                bsf            BUTTON,4                      ; button 4 was pressed
006E                 00479 ButtonEnd
006E   28A9          00480                goto           POP
                     00481
                     00482 ;*************************************************************************
                     00483 ; The !EOUT pin of the CS5460 outputs a pulse (active low) whenever a programmed
                     00484 ; amount of energy has been transferred between the line and the load. The !EDIR
                     00485 ; output indicates the direction of energy flow.
                     00486 ;
                     00487 ; The !EOUT pin drives the INT pin of the PIC16C923, causing an interrupt. This
                     00488 ; ISR examines the state of the !EOUT pin via RB1 to decide if WATTTMP<H:L>
                     00489 ; should be incremented (energy to the load) or decremented (energy from the load).
                     00490 ;
                     00491 ; When the WATTTMP<H:L> reaches 360 (3600 WattSeconds), WATTHR<H:L> is incremented,
                     00492 ; and WATTTMP<H:L> is cleared.  If the pulse count is decremented to less than 0,
                     00493 ; WATTTMP<H:L> is reset to 359 and WATTHR<H:L> is decremented.
                     00494 ;
                     00495 ; A PULSEC<H:L> counter is also incremented/decremented at the same time. This
                     00496 ; counter is used to count the number of pulses that occur each second for true power
                     00497 ; and power factor calculations.
                     00498 ;*************************************************************************
                     00499 ; Inputs:  Pins RB0 and RB1, PULSEC<H:L>, WATTTMP<H:L>, WATTHR<H:L>
                     00500 ; Outputs: PULSEC<H:L>, WATTTMP<H:L>, WATTHR<H:L>
                     00501 ;*************************************************************************
                     00502 ; Pulse from CS4560 !EOUT pin (active low)
                     00503 ; CS4560 is configured for
                     00504 ;   1 pulse  =   10 Watt*Sec
                     00505 ; 360 pulses = 3600 Watt*Sec = 1 Watt*Hr
                     00506
006F                 00507 RB0ISR
006F   108B          00508                bcf            INTCON,INTF    ; raw pulses per second from the CS4560
0070   1C86          00509                btfss          PORTB,1        ; Was EDIR high?
0071   2897          00510                goto           DirPlus        ; yes, increment counter
                     00511
0072                 00512 DirMinus
0072   03AA          00513                decf           PULSECL,F      ; Decrement pulse count
0073   082A          00514                movf           PULSECL,W
0074   3AFF          00515                xorlw          0xFF
0075   1D03          00516                btfss          STATUS,Z
0076   287C          00517                goto           DM1
0077   03A9          00518                decf           PULSECH,F
                     00519
0078   1FAA          00520                btfss          PULSECL,7
```

```
0079   287C           00521                goto          DM1
007A   01A9           00522                clrf          PULSECH
007B   01AA           00523                clrf          PULSECL
                      00524
007C   03A6           00525 DM1            decf          WATTTMPL,F     ; decrement pulse count LSB
007D   0826           00526                movf          WATTTMPL,W     ; get LSB
007E   3AFF           00527                xorlw         0xFF           ; test for underflow
007F   1D03           00528                btfss         STATUS,Z       ; was there an underflow?
0080   28A9           00529                goto          POP            ; no, continue
                      00530
0081   03A5           00531                decf          WATTTMPH,F     ; decrement pulse count MSB
0082   0825           00532                movf          WATTTMPH,W     ; get LSB
0083   3AFF           00533                xorlw         0xFF           ; test for underflow
0084   1D03           00534                btfss         STATUS,Z       ; was there an underflow?
0085   28A9           00535                goto          POP            ; no, continue
                      00536
0086   3001           00537                movlw         0x01           ; reset counter
0087   00A5           00538                movwf         WATTTMPH
0088   3067           00539                movlw         0x67
0089   00A6           00540                movwf         WATTTMPL
                      00541
008A   03A8           00542                decf          WATTHRL,F      ; decrement pulse count
008B   0828           00543                movf          WATTHRL,W      ; get LSB
008C   3AFF           00544                xorlw         0xFF           ; test for underflow
008D   1D03           00545                btfss         STATUS,Z       ; was there an underflow?
008E   28A9           00546                goto          POP            ; no, continue
                      00547
008F   03A7           00548                decf          WATTHRH,F      ; decrement pulse count
0090   0827           00549                movf          WATTHRH,W      ; get LSB
0091   3AFF           00550                xorlw         0xFF           ; test for underflow
0092   1D03           00551                btfss         STATUS,Z       ; was there an underflow?
0093   28A9           00552                goto          POP            ; no, continue
                      00553
0094   01A8           00554                clrf          WATTHRL        ; yes, do not decrement below zero
0095   01A7           00555                clrf          WATTHRH
                      00556
0096   28A9           00557                goto          POP            ; exit interrupt
                      00558
0097                  00559 DirPlus
0097   0FAA           00560                incfsz        PULSECL,F      ; increment pulse count
0098   289A           00561                goto          DP2
0099   0AA9           00562                incf          PULSECH,F
                      00563
009A   0FA6           00564 DP2            incfsz        WATTTMPL,F     ; increment pulse count
009B   289D           00565                goto          DP1
009C   0AA5           00566                incf          WATTTMPH,F     ; increment pulse count
```

```
                        00567
009D   1C25             00568 DP1           btfss           WATTTMPH,0      ;
009E   28A9             00569               goto            POP             ; not yet 360 pulses
                        00570
009F   0826             00571               movf            WATTTMPL,W      ; get pulse count MSB
00A0   3C67             00572               sublw           0x67            ; test for 360 pulses (256 + 104 = 360) (1WHr)
00A1   1803             00573               btfsc           STATUS,C        ; was there a borrow?
00A2   28A9             00574               goto            POP             ; no, count raw pulse
                        00575
00A3   01A6             00576               clrf            WATTTMPL        ; yes, clear counter
00A4   01A5             00577               clrf            WATTTMPH        ; yes, clear counter
                        00578
00A5   0FA8             00579               incfsz          WATTHRL,F       ; increment WattHOUR LSB
00A6   28A9             00580               goto            POP
00A7   0AA7             00581               incf            WATTHRH,F       ; increment WattHOUR MSB
                        00582
00A8   28A9             00583               goto            POP
                        00584
                        00585 ;*****************************************************************************
                        00586 ; Restore the values that were in the W, STATUS, PCLATH, and FSR
                        00587 ; registers just after the interrupt was called.
                        00588 ;
                        00589 ; These values were saved in the routine at the beginning of the
                        00590 ; program at "org 4 ; Interrupt vector"
                        00591
00A9                    00592 POP
00A9   0877             00593               movf            TEMPPCLATH,W
00AA   008A             00594               movwf           PCLATH
00AB   0876             00595               movf            TEMPFSR,W
00AC   0084             00596               movwf           FSR
00AD   0E79             00597               swapf           TEMPSTAT,W
00AE   0083             00598               movwf           STATUS
00AF   0EF8             00599               swapf           TEMPW,F
00B0   0E78             00600               swapf           TEMPW,W
00B1   0009             00601               retfie                          ; return from interrupt
                        00602
                        00603 ;*****************************************************************************
                        00604 ; Program start                                                               *
                        00605 ;*****************************************************************************
                        00606
00B2                    00607 Start
00B2   2494             00608               call            InitPeriph      ; initialize controller peripherals and some variables
                        00609
00B3   260B             00610               call            InitLCD         ; initialize the LCD module, setup registers
                        00611
00B4   2612             00612               call            ClrLCD          ; and clear the LCD display
```

```
                        00613
                        00614 ;***************************************************************************
                        00615 ; Write default calibration values in CAL.INC to the EEPROM.
                        00616 ; This is used to write calibration values for a new unit.
                        00617 ; Uncomment the next instruction to allow writing values to EEPROM memory.
                        00618 ;
                        00619 ;               call            WriteSer
                        00620 ;***************************************************************************
                        00621
00B5   222A             00622                 call            InitValues      ; get default values from the EEPROM,
                        00623                                                 ; place them in variables and write to CS5460,
                        00624                                                 ; initialize interrupts
                        00625
00B6   178B             00626                 bsf             INTCON,GIE      ; enable all interrupts
                        00627
00B7   0806             00628                 movf            PORTB,W         ; get states of BUTTONs (normally high, active low))
00B8   39F0             00629                 andlw           0xf0            ; mask off low 4 bits
00B9   1903             00630                 btfsc           STATUS,Z        ; are all four BUTTONs pressed?
00BA   21DD             00631                 call            ClearEEPROM     ; yes, clear the accumulated Watt HOURs from the EEPROM
                        00632                                                 ; and rewrite calibration values to CS5460
00BB                    00633 Continue2
00BB   0806             00634                 movf            PORTB,W         ; get states of buttons (normally high)
00BC   39F0             00635                 andlw           0xf0            ; mask off low 4 bits
00BD   3C90             00636                 sublw           0x90            ; center two buttons
00BE   1903             00637                 btfsc           STATUS,Z        ; are only center 2 buttons pressed
00BF   2CB5             00638                 goto            SetClock        ; yes, set the "Time of Day" clock
                        00639
00C0                    00640 Continue3
00C0   0806             00641                 movf            PORTB,W         ; get states of buttons (normally high)
00C1   39F0             00642                 andlw           0xf0            ; mask off low 4 bits
00C2   3C60             00643                 sublw           0x60            ; outside two buttons
00C3   1D03             00644                 btfss           STATUS,Z        ; are only outer 2 buttons pressed?
00C4   28C9             00645                 goto            Continue4       ; no, skip next few instructions
                        00646
00C5   3001             00647                 movlw           0x01
00C6   00B6             00648                 movwf           CALMODE         ; set to calibrate Voltage Offset
00C7   3037             00649                 movlw           CALDIG          ; set to digit 1 (left)
00C8   2B06             00650                 goto            Calibrate       ; Modify calibration registers in the CS4560
                        00651
                        00652 ;*******************************************************
                        00653 ; If no BUTTONs are pressed at reset
                        00654 ; the real program starts here
                        00655
00C9                    00656 Continue4
00C9   30E8             00657                 movlw           b'11101000'     ; CS5460 "start continuous conversions" command
00CA   00BC             00658                 movwf           TXDATA          ; Send the command
```

```
                    00659
00CB   22FD         00660               call          SSPCmd         ; send TXDATA only to CS5460
                    00661
00CC   225D         00662               call          Message        ; Display Startup Scrolling Message
                    00663                                            ; until a button is pressed
                    00664
00CD   28CE         00665               goto          TimeLoop       ; Then start executing the main loop begining
                    00666                                            ; with the Display time.
                    00667
                    00668 ;*****************************************************************************
                    00669 ; Program main loop
                    00670 ;*****************************************************************************
                    00671
                    00672
                    00673 ;************* Display Time on LCD ***************************************
                    00674 ; HOURs, MINUTEs, & SECONDs are counted in the Time Interrupt routine.
                    00675 ;
                    00676 ; The display is updated once after one second while in this loop before execution
                    00677 ; continues to ELoop.
                    00678 ;
                    00679 ; If any button is pressed, execution remains within this loop until 2 seconds
                    00680 ; after all buttons have been released.
                    00681 ;*****************************************************************************
                    00682 ; Inputs: UPDATE, MODEINC, PORTB
                    00683 ; Ouputs: none
                    00684
00CE                00685 TimeLoop
00CE   2295         00686               call          UpdateDisplay  ; displays current time on LCD
                    00687
00CF                00688 TLoop
00CF   1873         00689               btfsc         UPDATE,0                      ; 1 second passed?
00D0   2295         00690               call          UpdateDisplay  ; yes, displays current time on LCD.
                    00691
00D1   1FF2         00692               btfss         MODEINC,7                     ; 2 seconds yet?
00D2   28D5         00693               goto          TLoop2                        ; no, skip over jump
                    00694
00D3   01F2         00695               clrf          MODEINC                       ; yes, clear MODE counter
00D4   28DC         00696               goto          ELoop                         ; goto ELoop
00D5                00697 TLoop2
00D5   0806         00698               movf          PORTB,W                       ; check buttons
00D6   39F0         00699               andlw         0xf0                          ; mask off unneeded bits
00D7   3CF0         00700               sublw         0xf0                          ; compare with no pressed buttons state
00D8   1903         00701               btfsc         STATUS,Z                      ; buttons pressed?
00D9   28CF         00702               goto          TLoop                         ; no, wait some more
                    00703
00DA   01F2         00704               clrf          MODEINC                       ; yes, clear MODE counter
```

```
00DB   28CF          00705                goto          TLoop                          ; wait forever
                     00706
                     00707 ;************** Display Voltage on LCD ************************************
                     00708 ; Reads RMS voltage (as fraction of full scale, 0 <= VRMS < 1 ) from CS5460,
                     00709 ; multiplies by the full scale voltage to get actual voltage in 2 byte binary,
                     00710 ; converts binary to 5 digit (3 byte) BCD (Binary Coded Decimal).
                     00711 ;
                     00712 ; The display is blanked, an "E" is written to the leftmost digit, and the decimal
                     00713 ; point is turned on. The BCD result is displayed, skipping leading zeros in the
                     00714 ; 10's and 100's digits.
                     00715 ;
                     00716 ; If any button is pressed, execution remains within this loop until 2 seconds
                     00717 ; after all buttons have been released.
                     00718 ;*************************************************************************
                     00719 ; Inputs: MAXVOLT<H:L>, Vrms data from CS5460, UPDATE, MODEINC, PORTB
                     00720 ; Ouputs: none
                     00721
00DC                 00722 ELoop
00DC   3018          00723                movlw         b'00011000'                    ; read RMS Voltage from CS4560
00DD   00BC          00724                movwf         TXDATA
00DE   22BF          00725                call          SSPRead
                     00726
00DF   0839          00727                movf          RXDATA0,W                      ; move received Vrms data to
00E0   00C0          00728                movwf         AARGB0                         ; multiplicand
00E1   083A          00729                movf          RXDATA1,W
00E2   00C1          00730                movwf         AARGB1
                     00731
00E3   3000          00732                movlw         MAXVOLTH                       ; load full scale voltage to
00E4   00C5          00733                movwf         BARGB0                         ; multiplicand
00E5   30ED          00734                movlw         MAXVOLTL
00E6   00C6          00735                movwf         BARGB1
                     00736
00E7   25D1          00737                call          FXM1616U                       ; Math 16bit * 16bit multiply routine
                     00738
00E8   0841          00739                movf          AARGB1,W                       ; move result for Bin2BCD
00E9   00D2          00740                movwf         TEMPH                          ; this is the actual voltage in binary
00EA   0842          00741                movf          AARGB2,W
00EB   00D1          00742                movwf         TEMPL
                     00743
00EC   24DC          00744                call          Bin2BCD16                      ; Change 16 bit binary to 5 digit decimal
                     00745
00ED   2612          00746                call          ClrLCD                         ; clear display
                     00747
00EE   300E          00748                movlw         0x0E                           ; display "E" on LCD (for Volts)
00EF   261F          00749                call          LoadD1
00F0   1703          00750                bsf           STATUS,RP1                     ; Select Bank 2
```

```
00F1   1798            00751                bsf            DEC6                          ; turn on the decimal point
00F2   1303            00752                bcf            STATUS,RP1                    ; Select Bank 0
                       00753
00F3   084E            00754                movf           R0,W                          ; if "Hundreds" digit is 0, leave blank
00F4   390F            00755                andlw          0x0F
00F5   1903            00756                btfsc          STATUS,Z
00F6   28F9            00757                goto           E5LCD
00F7   270C            00758                call           WriteLCD4                     ; Display 5 digits of data as decimal
00F8   2900            00759                goto           WaitE1
                       00760
00F9   0E4F            00761 E5LCD          swapf          R1,W                          ; if "Tens" digit is zero, leave blank
00FA   390F            00762                andlw          0x0F
00FB   1903            00763                btfsc          STATUS,Z
00FC   28FF            00764                goto           E4LCD
00FD   270F            00765                call           WriteLCD5                     ; Display 4 digits of data as decimal
00FE   2900            00766                goto           WaitE1
                       00767
00FF   2712            00768 E4LCD          call           WriteLCD6                     ; Display 3 digits of data as decimal
                       00769
0100   1073            00770 WaitE1         bcf            UPDATE,0
0101                   00771 WaitE
0101   1FF2            00772                btfss          MODEINC,7                     ; 2 seconds yet?
0102   2905            00773                goto           WaitE2                        ; no, skip over jump
                       00774
0103   01F2            00775                clrf           MODEINC                       ; yes, clear MODE counter
0104   290E            00776                goto           ILoop                         ; go to the Current loop
                       00777
0105                   00778 WaitE2
0105   1873            00779                btfsc          UPDATE,0                      ; 1 second passed?
0106   28DC            00780                goto           ELoop                         ; yes, take new reading and display it.
                       00781
0107   0806            00782                movf           PORTB,W                       ; no, check buttons
0108   39F0            00783                andlw          0xf0                          ; mask off unneeded bits
0109   3CF0            00784                sublw          0xf0                          ; compare with no pressed button state
010A   1903            00785                btfsc          STATUS,Z                      ; button pressed?
010B   2901            00786                goto           WaitE                         ; no, wait some more
                       00787
010C   01F2            00788                clrf           MODEINC                       ; yes, clear MODE counter
010D   2901            00789                goto           WaitE                         ; wait some more
                       00790
                       00791 ;************* Display Current on LCD ************************************
                       00792 ; Reads RMS current (as fraction of full scale, 0 <= IRMS < 1 ) from CS5460,
                       00793 ; multiplies by the full scale current to get actual current in 2 byte binary,
                       00794 ; converts binary to 5 digit (3 byte) BCD (Binary Coded Decimal).
                       00795 ;
                       00796 ; The display is blanked, an "C" is written to the leftmost digit, and the decimal
```

```
                    00797 ; point is turned on. The BCD result is displayed, skipping the leading zero in the
                    00798 ; 10's digit.
                    00799 ;
                    00800 ; If any button is pressed, execution remains within this loop until 2 seconds
                    00801 ; after all buttons have been released.
                    00802 ;*****************************************************************************
                    00803 ; Inputs: MAXCURR<H:L>, Irms data from CS5460, UPDATE, MODEINC, PORTB
                    00804 ; Ouputs: none
                    00805
010E                00806 ILoop
010E  3016          00807               movlw       b'00010110'             ; read Irms from the CS4560
010F  00BC          00808               movwf       TXDATA
0110  22BF          00809               call        SSPRead
                    00810
0111  0839          00811               movf        RXDATA0,W               ; move received data to
0112  00C0          00812               movwf       AARGB0                  ; multiplicand
0113  083A          00813               movf        RXDATA1,W
0114  00C1          00814               movwf       AARGB1
                    00815
0115  3000          00816               movlw       MAXCURRH                ; move full scale Current value
0116  00C5          00817               movwf       BARGB0                  ; to multiplicand
0117  30FA          00818               movlw       MAXCURRL
0118  00C6          00819               movwf       BARGB1
                    00820
0119  25D1          00821               call        FXM1616U                ; Math 16bit * 16bit multiply routine
                    00822
011A  0841          00823               movf        AARGB1,W                ; move result for Bin2BCD
011B  00D2          00824               movwf       TEMPH                   ; this is actual current in binary
011C  0842          00825               movf        AARGB2,W
011D  00D1          00826               movwf       TEMPL
                    00827
011E  24DC          00828               call        Bin2BCD16               ; Change 16 bit binary to 5 digit decimal (BCD)
                    00829
011F  2612          00830               call        ClrLCD                  ; clear display
                    00831
0120  300C          00832               movlw       0x0C                    ; load W with char "C" to display
0121  261F          00833               call        LoadD1                  ; display "C" on LCD (for Current)
0122  1703          00834               bsf         STATUS,RP1              ; Select Bank 2
0123  1519          00835               bsf         DEC5                    ; turn on the decimal point
0124  1303          00836               bcf         STATUS,RP1              ; Select Bank 0
                    00837
0125  084E          00838               movf        R0,W                    ; if "tens" digit is 0, leave blank
0126  390F          00839               andlw       0x0F
0127  1903          00840               btfsc       STATUS,Z
0128  292B          00841               goto        C5LCD
0129  270C          00842               call        WriteLCD4               ; Display 5 digits of data as decimal
```

```
012A   292C           00843                 goto          WaitC1
012B   270F           00844 C5LCD           call          WriteLCD5                ; Display 4 digits of data as decimal
                      00845
012C   1073           00846 WaitC1          bcf           UPDATE,0
012D                  00847 WaitC
012D   1FF2           00848                 btfss         MODEINC,7                ; 2 seconds yet?
012E   2931           00849                 goto          WaitC2                   ; no, skip over jump
                      00850
012F   01F2           00851                 clrf          MODEINC                  ; yes, clear MODE counter
0130   293A           00852                 goto          APLoop                   ; go to the Apparent Power loop
                      00853
0131                  00854 WaitC2
0131   1873           00855                 btfsc         UPDATE,0                 ; 1 second passed?
0132   290E           00856                 goto          ILoop                    ; yes, take new reading and display it.
0133   0806           00857                 movf          PORTB,W                  ; no, check buttons
0134   39F0           00858                 andlw         0xf0                     ; mask off unneeded bits
0135   3CF0           00859                 sublw         0xf0                     ; compare with no pressed button state
0136   1903           00860                 btfsc         STATUS,Z                 ; button pressed?
0137   292D           00861                 goto          WaitC                    ; no, wait some more
                      00862
0138   01F2           00863                 clrf          MODEINC                  ; yes, clear MODE counter
0139   292D           00864                 goto          WaitC                    ; wait some more
                      00865
                      00866
                      00867 ;************* Display Apparent Power on LCD ************************************
                      00868 ; Calls routine to calculate apparent power, converts the binary result to BCD, clears
                      00869 ; the display, displays "AP", and calls the subroutine that displays the power result.
                      00870 ;
                      00871 ; If any button is pressed, execution remains within this loop until 2 seconds
                      00872 ; after all buttons have been released.
                      00873 ;*******************************************************************************
                      00874 ; Inputs: UPDATE, MODEINC, PORTB
                      00875 ; Ouputs: none
                      00876
013A                  00877 APLoop
013A   24FF           00878                 call          CalcAP                   ; get Vrms, Irms, multiply, save as APH:APL(binary)
                      00879
013B   082D           00880                 movf          APH,W                    ; get Apparent Power high byte
013C   00D2           00881                 movwf         TEMPH
                      00882
013D   082E           00883                 movf          APL,W                    ; get Apparent Power low byte
013E   00D1           00884                 movwf         TEMPL
                      00885
013F   24DC           00886                 call          Bin2BCD16                ; convert binary to decimal
                      00887
0140   2612           00888                 call          ClrLCD                   ; clear the LCD display
```

```
                        00889
0141    300A            00890                   movlw           0x0A                    ; A
0142    261F            00891                   call            LoadD1
0143    3010            00892                   movlw           0x10                    ; P
0144    263A            00893                   call            LoadD2
                        00894
0145    271C            00895                   call            DispPwr                 ; display power with lead zero blanking
                        00896
0146    1073            00897 WaitAP1           bcf             UPDATE,0
                        00898
0147                    00899 WaitAP
0147    1FF2            00900                   btfss           MODEINC,7               ; 2 seconds yet?
0148    294B            00901                   goto            WaitAP2                 ; no, skip over jump
                        00902
0149    01F2            00903                   clrf            MODEINC                 ; yes, clear MODE counter
014A    2954            00904                   goto            TPLoop                  ; goto True Power Loop
                        00905
014B                    00906 WaitAP2
014B    1873            00907                   btfsc           UPDATE,0                ; 1 second passed?
014C    293A            00908                   goto            APLoop                  ; yes, display new Apparent Power.
                        00909
014D    0806            00910                   movf            PORTB,W                 ; check buttons
014E    39F0            00911                   andlw           0xf0                    ; mask off unneeded bits
014F    3CF0            00912                   sublw           0xf0                    ; compare with no pressed button state
0150    1903            00913                   btfsc           STATUS,Z                ; button pressed?
0151    2947            00914                   goto            WaitAP                  ; no, wait some more
                        00915
0152    01F2            00916                   clrf            MODEINC                 ; yes, clear MODE counter
0153    2947            00917                   goto            WaitAP                  ; wait some more
                        00918
                        00919 ;************* Display True Power on LCD ****************************************
                        00920 ; Calls routine to calculate true power, converts the binary result to BCD, clears
                        00921 ; the display, displays "tP", and calls the subroutine that displays the power result.
                        00922 ;
                        00923 ; If any button is pressed, execution remains within this loop until 2 seconds
                        00924 ; after all buttons have been released.
                        00925 ;*******************************************************************************
                        00926 ; Inputs: UPDATE, MODEINC, PORTB
                        00927 ; Ouputs: none
                        00928
0154                    00929 TPLoop
0154    2518            00930                   call            CalcTP                  ; calc True Power, Result in TPH:TPL in Watts as binary
                        00931
0155    082F            00932                   movf            TPH,W                   ; Pulse per second MSB
0156    00D2            00933                   movwf           TEMPH
0157    0830            00934                   movf            TPL,W                   ; Pulse per second LSB
```

```
0158   00D1            00935                movwf          TEMPL
                       00936
0159   24DC            00937                call           Bin2BCD16                    ; 16 bit binary to BCD routine
                       00938
015A   2612            00939                call           ClrLCD
                       00940
015B   3013            00941                movlw          0x13                         ; t
015C   261F            00942                call           LoadD1
015D   3010            00943                movlw          0x10                         ; P
015E   263A            00944                call           LoadD2
                       00945
015F   271C            00946                call           DispPwr                      ; display power with lead zero blanking
                       00947
0160   1073            00948                bcf            UPDATE,0
0161                   00949 WaitTP
0161   1FF2            00950                btfss          MODEINC,7                    ; 2 seconds yet?
0162   2965            00951                goto           WaitTP2                      ; no, skip over jump
                       00952
0163   01F2            00953                clrf           MODEINC                      ; yes, clear MODE counter
0164   296E            00954                goto           PFLoop                       ; goto Power Factor loop
                       00955
0165                   00956 WaitTP2
0165   1873            00957                btfsc          UPDATE,0                     ; 1 second passed?
0166   2954            00958                goto           TPLoop
                       00959
0167   0806            00960                movf           PORTB,W                      ; check buttons
0168   39F0            00961                andlw          0xf0                         ; mask off unneeded bits
0169   3CF0            00962                sublw          0xf0                         ; compare with no pressed button state
016A   1903            00963                btfsc          STATUS,Z                     ; button pressed?
016B   2961            00964                goto           WaitTP                       ; no, wait some more
                       00965
016C   01F2            00966                clrf           MODEINC                      ; yes, clear MODE counter
016D   2961            00967                goto           WaitTP                       ; wait some more
                       00968
                       00969
                       00970 ;************* Display Power Factor on LCD ****************************************
                       00971 ; Calls routines to calculate apparent power APH:APL and true power TPH:TPL.
                       00972 ; If APH:APL=0, then PF=1.000 is reported.
                       00973 ;
                       00974 ; The power factor is calculated by dividing the true power by the apparent power.
                       00975 ; The binary result is converted to BCD.
                       00976 ;
                       00977 ; The display is blanked, "tP" is displayed, and the PF is displayed starting with the 1's digit.
                       00978 ;
                       00979 ; If any button is pressed, execution remains within this loop until 2 seconds
                       00980 ; after all buttons have been released.
```

```
                    00981 ;*****************************************************************************
                    00982 ; Inputs: UPDATE, MODEINC, PORTB
                    00983 ; Ouputs: none
                    00984 ;*****************************************************************************
                    00985 ; PF = TP / AP      TP <= AP,   0<=PF<=1
                    00986 ;
                    00987 ; If AP=0, then a divide by zero condition exists, report back PF=1.000
                    00988 ; This is a no load condition. Vrms cannot be zero (this device wont work). Irms is zero.
                    00989 ;
                    00990 ; Due to delays in the CS5460, PF can sometimes compute to greater than 1.000. This
                    00991 ; is an incorrect result. In such cases, limit PF to 1.000.
                    00992 ; The next calculation will report the correct PF.
                    00993 ; This usually occurs only when the load charactoristics are rapdily changing.
                    00994 ;*****************************************************************************
                    00995
016E                00996 PFLoop
016E   24FF          00997               call          CalcAP                      ; calculate present Aparent Power
                    00998
                    00999               ; checking for divide by zero condition ( APH:APL=0 )
016F   08AD          01000               movf          APH,F                       ; test APH for non-zero value
0170   1D03          01001               btfss         STATUS,Z                    ; was APH zero?
0171   297B          01002               goto          PFNZero                     ; APH is non-zero, PF can be calculated
                    01003
0172   082E          01004               movf          APL,W                       ; APL may be very close to zero
0173   39F0          01005               andlw         0xF0                        ; discard low 4 bits
0174   1D03          01006               btfss         STATUS,Z                    ; is APL>0x0F?
0175   297B          01007               goto          PFNZero                     ; APL is non-zero
                    01008
0176   3003          01009 PFUnity       movlw         0x03                        ; report PF=1.000
0177   00D2          01010               movwf         TEMPH
0178   30E8          01011               movlw         0xE8
0179   00D1          01012               movwf         TEMPL
                    01013
017A   2996          01014               goto          PFZero                      ; jump directly to display routine
                    01015
017B   2518          01016 PFNZero       call          CalcTP                      ; calculate present True Power
                    01017                                                         ; save results in TPH:TPL as Watts in binary
                    01018
017C   082F          01019               movf          TPH,W                       ; load True Power
017D   00C0          01020               movwf         AARGB0
017E   0830          01021               movf          TPL,W
017F   00C1          01022               movwf         AARGB1
0180   01C2          01023               clrf          AARGB2
                    01024
0181   082D          01025               movf          APH,W                       ; load Apparent Power
0182   00C5          01026               movwf         BARGB0
```

```
0183   082E           01027                movf           APL,W
0184   00C6           01028                movwf          BARGB1
                      01029
0185   2525           01030                call           FXD2416U                    ; divide True Power by Apparent Power
                      01031
0186   08C1           01032                movf           AARGB1,F                    ; Test AARGB1
0187   1D03           01033                btfss          STATUS,Z                    ; is AARGB1 non-zero?
0188   2976           01034                goto           PFUnity                     ; yes, report PF => 1.000
                      01035
0189   0841           01036                movf           AARGB1,W                    ; move result for scaling
018A   00C0           01037                movwf          AARGB0
018B   0842           01038                movf           AARGB2,W
018C   00C1           01039                movwf          AARGB1
                      01040
018D   3003           01041                movlw          0x03                        ; 1000 decimal
018E   00C5           01042                movwf          BARGB0
018F   30E8           01043                movlw          0xE8
0190   00C6           01044                movwf          BARGB1
                      01045
0191   25D1           01046                call           FXM1616U                    ; multiply result by 1000 decimal
                      01047
0192   0841           01048                movf           AARGB1,W
0193   00D2           01049                movwf          TEMPH
0194   0842           01050                movf           AARGB2,W
0195   00D1           01051                movwf          TEMPL
                      01052
0196                  01053 PFZero         ; put HEX value in TEMPH:TEMPL (MSB:LSB) and BCD is returned in R0:R1:R2
0196   24DC           01054                call           Bin2BCD16                   ; 16 bit binary to BCD routine
                      01055
0197   2612           01056                call           ClrLCD
                      01057
0198   3010           01058                movlw          0x10                        ; P
0199   261F           01059                call           LoadD1
019A   300F           01060                movlw          0x0F                        ; F
019B   263A           01061                call           LoadD2
                      01062
019C   1283 1703      01063                banksel        LCDD02
019E   1519           01064                bsf            DEC5                        ; turn on the decimal point
019F   1283 1303      01065                banksel        PORTA
                      01066
01A1   270F           01067                call           WriteLCD5                   ; Write the data as formatted decimal
                      01068
01A2   1073           01069                bcf            UPDATE,0
                      01070
01A3                  01071 WaitPF
01A3   1FF2           01072                btfss          MODEINC,7                   ; 2 seconds yet?
```

```
01A4   29A7          01073                goto          WaitPF2                    ; no, skip over jump
                     01074
01A5   01F2          01075                clrf          MODEINC                    ; yes, clear MODE counter
01A6   29B0          01076                goto          WHLoop                     ; goto Kwhr loop
                     01077
01A7                 01078 WaitPF2
01A7   1873          01079                btfsc         UPDATE,0                   ; 1 second passed?
01A8   296E          01080                goto          PFLoop                     ; go to top of Power Factor loop
                     01081
01A9   0806          01082                movf          PORTB,W                    ; check BUTTONs
01AA   39F0          01083                andlw         0xf0                       ; mask off unneeded bits
01AB   3CF0          01084                sublw         0xf0                       ; compare with no pressed BUTTONs state
01AC   1903          01085                btfsc         STATUS,Z                   ; BUTTON pressed?
01AD   29A3          01086                goto          WaitPF                     ; no, wait some more
                     01087
01AE   01F2          01088                clrf          MODEINC                    ; yes, clear MODE counter
01AF   29A3          01089                goto          WaitPF                     ; wait some more
                     01090
                     01091 ;************* Display Kw/Hr on LCD ************************************
                     01092 ; The !EOUT pin of the CS4560 goes to the RB0/INT pin of the PIC16C923.
                     01093 ; The !EDIR pin of the CS5460 goes to the RB1 pin.
                     01094 ; The interrupt service routine RB0ISR counts the pulses and increments WATTHRH:WATTHRL.
                     01095 ;
                     01096 ; This routine takes the total energy in WATTHRH:WATTHRL, converts it to BCD, clears the display,
                     01097 ; displays "Hr", and calls the subroutine that displays power while blanking leading zeros.
                     01098 ;
                     01099 ; If any button is pressed, execution remains within this loop until 2 seconds
                     01100 ; after all buttons have been released.
                     01101 ;*********************************************************************
                     01102 ; Inputs: UPDATE, MODEINC, PORTB, WATTHRH:WATTHRL
                     01103 ; Ouputs: none
                     01104
01B0                 01105 WHLoop
01B0   2612          01106                call          ClrLCD
01B1   0828          01107                movf          WATTHRL,W      ; LSB of Watt hours
01B2   00D1          01108                movwf         TEMPL
01B3   0827          01109                movf          WATTHRH,W      ; MSB of Watt hours
01B4   00D2          01110                movwf         TEMPH
                     01111                                             ; put HEX value in TEMPL (LSB) and TEMPH (MSB),
01B5   24DC          01112                call          Bin2BCD16      ; BCD is returned in R2 (LSB) R1 and R0 (MSB)
                     01113                                             ; 16 bit binary to BCD routine
                     01114
01B6   2612          01115                call          ClrLCD         ; blank display
                     01116
                     01117 ;              Write a "Hr" to the first two digits (left side)
01B7   3018          01118                movlw         0x18           ; H
```

```
01B8   261F          01119                call          LoadD1
01B9   3014          01120                movlw         0x14          ; r
01BA   263A          01121                call          LoadD2
                     01122
01BB   271C          01123                call          DispPwr       ; display result without leading zeros
                     01124
01BC   1073          01125                bcf           UPDATE,0
01BD                 01126 WaitWH
01BD   1FF2          01127                btfss         MODEINC,7     ; 2 seconds yet?
01BE   29C1          01128                goto          WaitWH2       ; no, skip over jump
                     01129
01BF   01F2          01130                clrf          MODEINC       ; yes, clear MODE counter
01C0   28CE          01131                goto          TimeLoop      ; goto TimeLoop
                     01132
01C1                 01133 WaitWH2
01C1   1873          01134                btfsc         UPDATE,0      ; 1 second passed?
01C2   29B0          01135                goto          WHLoop        ; yes, take new reading and display it
                     01136
01C3   0806          01137                movf          PORTB,W       ; no, check BUTTONs
01C4   39F0          01138                andlw         0xf0          ; mask off unneeded bits
01C5   3CF0          01139                sublw         0xf0          ; compare with no pressed BUTTONs state
01C6   1903          01140                btfsc         STATUS,Z      ; BUTTON pressed?
01C7   29BD          01141                goto          WaitWH        ; no, wait some more
                     01142
01C8   01F2          01143                clrf          MODEINC       ; yes, clear MODE counter
01C9   29BD          01144                goto          WaitWH        ; wait some more
                     01145
                     01146
                     01147 ;*****************************************************************************
                     01148 ;              initialization routines
                     01149 ;*****************************************************************************
                     01150 ; Commands the CS5460 to perform a reset cycle via the SPI code.
                     01151 ; After the CS5460 is resets, the serial port is resynchronized by sending SYNC bytes.
                     01152 ;*****************************************************************************
                     01153 ; Inputs: none
                     01154 ; outputs: none
                     01155
01CA                 01156 ResetCS4560    ; all default values except bit 7 which starts a reset cycle
                     01157
01CA   3040          01158                movlw         b'01000000'   ; write to CS4560 Configuration Register
01CB   00BC          01159                movwf         TXDATA        ; command byte
01CC   3000          01160                movlw         b'00000000'
01CD   00BD          01161                movwf         TXDATA0       ; bits 23-16
01CE   3000          01162                movlw         b'00000000'
01CF   00BE          01163                movwf         TXDATA1       ; bits 15-8
01D0   3081          01164                movlw         b'10000001'   ; set bit 7 to Start a CS5460 Reset cycle
```

```
01D1   00BF          01165                movwf          TXDATA2        ; bits 7-0
01D2   22E1          01166                call           SSPWrite       ; send command to CS5460
                     01167
                     01168 ;*****************************************************************************
                     01169 ; Initialize the CS5460 Serial Port
                     01170 ; Write SYNC1 (0xFF) to the CS4560 up to 3 times followed by
                     01171 ; a SYNC0 (0xFE) to initialize the Serial Port
                     01172
01D3   30FF          01173                movlw          SYNC1          ; SYNC1 byte (0xFF)
01D4   00BC          01174                movwf          TXDATA
01D5   30FF          01175                movlw          SYNC1          ; SYNC1 byte (0xFF)
01D6   00BD          01176                movwf          TXDATA0
01D7   30FF          01177                movlw          SYNC1          ; SYNC1 byte (0xFF)
01D8   00BE          01178                movwf          TXDATA1
01D9   30FE          01179                movlw          SYNC0          ; SYNC0 byte (0xFE)
01DA   00BF          01180                movwf          TXDATA2
01DB   22E1          01181                call           SSPWrite       ; send sync bytes to CS5460
                     01182
01DC   0008          01183                return
                     01184
                     01185 ;*****************************************************************************
                     01186 ; This routine clears the accumulated Watt Hours
                     01187 ;
                     01188 ; The CS5460 is reset, the total energy is zeroed in the 24C01 EEPROM, and all
                     01189 ; internal variables that track accumlated energy are cleared. The display is
                     01190 ; blanked, and "CLEAr" is written to it.
                     01191 ;
                     01192 ; Calibration values are read from the 24C01 and written back to the CS5460.
                     01193 ;*****************************************************************************
                     01194 ; Inputs: none
                     01195 ; outputs: none
                     01196
01DD                 01197 ClearEEPROM
01DD   21CA          01198                call           ResetCS4560
                     01199
                     01200 ; Write "00" to the EEPROM addresses
                     01201
01DE   01D6          01202                clrf           EEDATA         ; set data to 0
01DF   300D          01203                movlw          0x0D           ; EE address of Watt HOUR MSB
01E0   00D5          01204                movwf          EEADDR
                     01205
01E1   241A          01206                call           EEWrite        ; clears WHr MSB
                     01207
01E2   0AD5          01208                incf           EEADDR,F       ; clears WHr LSB
01E3   241A          01209                call           EEWrite
                     01210
```

```
01E4   01A8           01211                clrf           WATTHRL        ; clear total energy
01E5   01A7           01212                clrf           WATTHRH
01E6   01A5           01213                clrf           WATTTMPH       ; clear pulse counter
01E7   01A6           01214                clrf           WATTTMPL
01E8   01B8           01215                clrf           UPDATEWH       ; clear 8 minute timer
                      01216
01E9   21F9           01217                call           GetDefaults    ; Put the default Voltage and Current
                      01218                                               ; Offset and Gain values back in the CS4560
01EA   2612           01219                call           ClrLCD
01EB   300C           01220                movlw          0x0C           ; "C"          ; Write CLEAr to the LCD
01EC   263A           01221                call           LoadD2
01ED   301D           01222                movlw          0x1D           ; "L"
01EE   2655           01223                call           LoadD3
01EF   300E           01224                movlw          0x0E           ; "E"
01F0   2670           01225                call           LoadD4
01F1   300A           01226                movlw          0x0A           ; "A"
01F2   268B           01227                call           LoadD5
01F3   3014           01228                movlw          0x14           ; "r"
01F4   26A6           01229                call           LoadD6
                      01230
01F5                  01231 ClearWait      ; wait for BUTTON press to continue
                      01232
01F5   1C74           01233                btfss          BUTTON,0
01F6   29F5           01234                goto           ClearWait
01F7   01F4           01235                clrf           BUTTON         ; clear button state
01F8   0008           01236                return
                      01237
                      01238 ;*************************************************************************
                      01239 ; This routine fetches the calibration values from the EEPROM and writes them to the
                      01240 ; CS4560 registers. The calibration data consists of
                      01241 ; Voltage Offset, Current Offset, Voltage Gain, Current Gain and Pulse Rate Gain
                      01242 ;
                      01243 ; It also fetches the accumulated Watt HOURs from the EEPROM and writes it to the variables in RAM.
                      01244 ;
                      01245 ; This routine is called at power-up.
                      01246 ;*************************************************************************
                      01247 ; Inputs: none
                      01248 ; outputs: WATTHRH:WATTHRL
                      01249
                      01250
01F9                  01251 GetDefaults
                      01252 ; gets 2 bytes of WATTHR data from EEPROM, stores in WATTHRH:WATTHRL
01F9   300D           01253                movlw          0x0D           ; address of Watt HOUR MSB in EEPROM
01FA   00D5           01254                movwf          EEADDR
01FB   242A           01255                call           EERead         ; get byte from address in EEPROM
01FC   0856           01256                movf           EEDATA,W       ; retrieve data
```

```
01FD   00A8          01257                movwf          WATTHRL        ; store in RAM
                     01258
01FE   0AD5          01259                incf           EEADDR,F       ; next higher EEPROM address
01FF   242A          01260                call           EERead         ; get byte from address in EEPROM
0200   0856          01261                movf           EEDATA,W       ; retrieve data
0201   00A7          01262                movwf          WATTHRH        ; store in RAM
                     01263
0202   01B8          01264                clrf           UPDATEWH
                     01265
                     01266 ; Voltage Gain
0203   3048          01267                movlw          b'01001000'    ; write to CS5460 Voltage Gain register
0204   00BC          01268                movwf          TXDATA         ; put in TX buffer
                     01269
0205   3007          01270                movlw          0x07           ; address in EEPROM for Voltage Gain MSB
0206   00D5          01271                movwf          EEADDR
                     01272
0207   221D          01273                call           Init5460       ; get voltage gain constant from EEPROM,
                     01274                                               ; write to corresponding registers in CS5460
                     01275 ; Current Gain
0208   3044          01276                movlw          b'01000100'    ; write to CS5460 Current Gain register
0209   00BC          01277                movwf          TXDATA
                     01278
020A   300A          01279                movlw          0x0A           ; address in EEPROM for Current Gain MSB
020B   00D5          01280                movwf          EEADDR
                     01281
020C   221D          01282                call           Init5460       ; get current gain constant from EEPROM,
                     01283                                               ; write to corresponding registers in CS5460
                     01284 ; Voltage Offset
020D   3046          01285                movlw          b'01000110'    ; write to CS5460 Voltage offset register
020E   00BC          01286                movwf          TXDATA         ; store in TX buffer
                     01287
020F   3001          01288                movlw          0x01           ; address of Voltage Offset MSB in EEPROM
0210   00D5          01289                movwf          EEADDR
                     01290
0211   221D          01291                call           Init5460       ; get voltage offset constant from EEPROM,
                     01292                                               ; write to corresponding registers in CS5460
                     01293 ; Current Offset
0212   3042          01294                movlw          b'01000010'    ; write to CS5460 Current offset register
0213   00BC          01295                movwf          TXDATA         ; store in TX buffer
                     01296
0214   3004          01297                movlw          0x04           ; address in EEPROM for Current Offset MSB
0215   00D5          01298                movwf          EEADDR
                     01299
0216   221D          01300                call           Init5460       ; get current offset constant from EEPROM,
                     01301                                               ; write to corresponding registers in CS5460
                     01302 ; Pulse Rate
```

```
0217   304C            01303                movlw          b'01001100'    ; write to CS5460 Pulse Rate register
0218   00BC            01304                movwf          TXDATA         ; store in TX buffer
                       01305
0219   300F            01306                movlw          0x0F           ; address in EEPROM for Pulse Rate MSB
021A   00D5            01307                movwf          EEADDR
                       01308
021B   221D            01309                call           Init5460       ; get Pulse Rate constant from EEPROM,
                       01310                                              ; write to corresponding registers in CS5460
021C   0008            01311                return
                       01312
                       01313 ;*****************************************************************************
                       01314 ; Reads 3 bytes from EEPROM, puts them into transmit buffer
                       01315 ; Sends CS5460 command and transmit buffer to CS5460
                       01316 ;*****************************************************************************
                       01317 ; Inputs:  TXDATA   CS5460 command
                       01318 ;          EEADDR   Address in EEPROM
                       01319 ; Outputs: none
                       01320
021D                   01321 Init5460
021D   242A            01322                call           EERead         ; get byte from address in EEPROM
021E   0856            01323                movf           EEDATA,W       ; retreive data
021F   00BD            01324                movwf          TXDATA0        ; store in TX buffer
                       01325
0220   0AD5            01326                incf           EEADDR,F       ; point to next address
0221   242A            01327                call           EERead         ; get byte from address in EEPROM
0222   0856            01328                movf           EEDATA,W       ; retreive data
0223   00BE            01329                movwf          TXDATA1        ; store in TX buffer
                       01330
0224   0AD5            01331                incf           EEADDR,F       ; point to next address
0225   242A            01332                call           EERead         ; get byte from address in EEPROM
0226   0856            01333                movf           EEDATA,W       ; retreive data
0227   00BF            01334                movwf          TXDATA2        ; store in TX buffer
                       01335
0228   22E1            01336                call           SSPWrite       ; send TX buffer to CS5460
                       01337
0229   0008            01338                return
                       01339
                       01340 ;*****************************************************************************
                       01341 ; See if this is a "warm" start or a "cold" start
                       01342 ;
                       01343 ; A "warm" start has occurred when the locations PWRUPAA and PWRUP55 contain 0xAA and 0x55, respectively
                             .
                       01344 ; A "cold" start has occurred when the locations PWRUPAA and PWRUP55 contain random data.
                       01345 ;
                       01346 ; If a cold start has occurred, the warm start data is written to PWRUPAA and PWRUP55, memory is cleared,
                       01347 ; the CS5460 is reset and calibration data is rewritten to it.
```

```
                    01348 ;
                    01349 ; Both warm and cold starts retrieve the device serial number and process it for display.
                    01350 ; All pending interrupts are cleared, and the PIC16C923 interrupts are initialized.
                    01351 ;*****************************************************************************
                    01352 ; Inputs: PWRUPAA, PWRUP55
                    01353 ; Outputs: PWRUPAA, PWRUP55, SERNUM, most memory may be cleared, INTCON, PIR1, PIE1
                    01354
022A                01355 InitValues                                      ; Read EEPROM default values
022A   0858          01356                movf           PWRUP55,W       ; get byte from RAM
022B   3C55          01357                sublw          0x55            ; compare with expected value
022C   1D03          01358                btfss          STATUS,Z        ; was there a match?
022D   2A32          01359                goto           ClearTime       ; no, was cold start.
                    01360
022E   0859          01361                movf           PWRUPAA,W       ; yes, might be a warm start.
022F   3CAA          01362                sublw          0xAA            ; if PWRUPAA <> 0xAA and PWRUP55
0230   1903          01363                btfsc          STATUS,Z        ; <> 0x55 then this was a cold start
0231   2A42          01364                goto           ReadSerNum      ; warm start
                    01365
0232                01366 ClearTime                                       ; This was a cold start, clear memory
0232   3020          01367                movlw          0x20            ; start of GPRs
0233   0084          01368                movwf          FSR
                    01369
0234                01370 ClrMem
0234   0180          01371                clrf           INDF            ; clear GPR
0235   0A84          01372                incf           FSR,F           ; increment POINTER
0236   0804          01373                movf           FSR,W           ; get POINTER
0237   3A80          01374                xorlw          0x80            ; test for first GPR to not clear
0238   1D03          01375                btfss          STATUS,Z        ; was there a match?
0239   2A34          01376                goto           ClrMem          ; no, clear another GPR
                    01377
023A   30AA          01378                movlw          0xAA            ; Write 0xAA to PWRUPAA
023B   00D9          01379                movwf          PWRUPAA
023C   3055          01380                movlw          0x55            ; Write 0x55 to PWRUP55
023D   00D8          01381                movwf          PWRUP55
                    01382
023E   301F          01383                movlw          0x1F
023F   00B3          01384                movwf          POINTER         ; initialize message pointer
                    01385
0240   21CA          01386                call           ResetCS4560     ; reset CS4560
0241   21F9          01387                call           GetDefaults ; get calibration data from EEPROM, write tp CS5460
                    01388
0242                01389 ReadSerNum                                      ; get serial number from EEPROM
0242   01D5          01390                clrf           EEADDR          ; EEPROM address, Serial Number at 0x00
0243   242A          01391                call           EERead          ; read the unit serial number byte
0244   0856          01392                movf           EEDATA,W        ; retreive EEPROM data
0245   00B5          01393                movwf          SERNUM          ; move to variable
```

```
0246   1903           01394                btfsc          STATUS,Z       ; if serial # =  0, no data
0247   2A50           01395                goto           ConfigInt
                      01396
0248   0835           01397                movf           SERNUM,W
0249   3CFF           01398                sublw          0xff
024A   1903           01399                btfsc          STATUS,Z       ; if serial # = FF, no data
024B   2A50           01400                goto           ConfigInt
                      01401
024C   0835           01402                movf           SERNUM,W       ; get device serial number
024D   00F0           01403                movwf          LCDTEMP1       ; store in temporary variable
024E   27B0           01404                call           Get7SegDat     ; get segment data
024F   00B5           01405                movwf          SERNUM         ; change to "display data"
                      01406
0250                  01407 ConfigInt      ; configure interrupts
0250   0806           01408                movf           PORTB,W        ; required to clear interrupt on PORTB change
0251   018C           01409                clrf           PIR1           ; clear all pending peripheral interrupts
0252   018B           01410                clrf           INTCON         ; clear all remaining interrupts and
                      01411                                              ; some interrupt enable flags
                      01412
0253   1683           01413                bsf            STATUS,RP0     ; select page 1
                      01414
0254   30BF           01415                movlw          b'10111111'    ; Interrupt on RB0/INT falling edge
0255   0081           01416                movwf          OPTION_REG
                      01417
0256   018C           01418                clrf           PIE1
                      01419 #IFDEF TMR1OSC
                      01420                bsf            PIE1,TMR1IE    ; enable Timer1 interrupt
                      01421 #ELSE
0257   150C           01422                bsf            PIE1,CCP1IE    ; enable CCP1 interrupt
                      01423 #ENDIF
0258   1283           01424                bcf            STATUS,RP0     ; select page 0
                      01425
0259   160B           01426                bsf            INTCON,INTE    ; enable RB0/INT pin interrupt
025A   158B           01427                bsf            INTCON,RBIE    ; enable change on PORTB interrupt
025B   170B           01428                bsf            INTCON,PEIE    ; enable peripheral interrupts
                      01429
025C   0008           01430                return
                      01431
                      01432 ;*****************************************************************************
                      01433 ; Displays the initial scrolling message                                     *
                      01434 ; (PIC16C923 WAtt HOur MEtEr "ser #")                                        *
                      01435 ;*****************************************************************************
                      01436 ; Inputs: BUTTON, UPDATE, POINTER
                      01437 ; Outputs: LCD display
                      01438
025D                  01439 Message
```

```
025D   0833          01440                movf          POINTER,W      ; POINTER is the charactor in the string
025E   00B4          01441                movwf         PTRTMP         ; we are going to display,
                     01442                                             ; save to temporary variable
                     01443
025F   27D6          01444                call          Msg            ; Call the table to get offset into Get7SegDat
0260   261F          01445                call          LoadD1         ; LCD position 1
0261   0BB4          01446                decfsz        PTRTMP,F
0262   2A65          01447                goto          Disp2          ; next character
0263   301F          01448                movlw         .31            ; at end of message, start over
0264   00B4          01449                movwf         PTRTMP
0265                 01450 Disp2
0265   27D6          01451                call          Msg
0266   263A          01452                call          LoadD2         ; LCD position 2
0267   0BB4          01453                decfsz        PTRTMP,F
0268   2A6B          01454                goto          Disp3
0269   301F          01455                movlw         .31
026A   00B4          01456                movwf         PTRTMP
026B                 01457 Disp3
026B   27D6          01458                call          Msg
026C   2655          01459                call          LoadD3         ; LCD position 3
026D   0BB4          01460                decfsz        PTRTMP,F
026E   2A71          01461                goto          Disp4
026F   301F          01462                movlw         .31
0270   00B4          01463                movwf         PTRTMP
0271                 01464 Disp4
0271   27D6          01465                call          Msg
0272   2670          01466                call          LoadD4         ; LCD position 4
0273   0BB4          01467                decfsz        PTRTMP,F
0274   2A77          01468                goto          Disp5
0275   301F          01469                movlw         .31
0276   00B4          01470                movwf         PTRTMP
0277                 01471 Disp5
0277   27D6          01472                call          Msg
0278   268B          01473                call          LoadD5         ; LCD position 5
0279   0BB4          01474                decfsz        PTRTMP,F
027A   2A7D          01475                goto          Disp6
027B   301F          01476                movlw         .31
027C   00B4          01477                movwf         PTRTMP
027D                 01478 Disp6
027D   27D6          01479                call          Msg
027E   26A6          01480                call          LoadD6         ; LCD position 6
027F   0BB4          01481                decfsz        PTRTMP,F
0280   2A83          01482                goto          Disp7
0281   301F          01483                movlw         .31
0282   00B4          01484                movwf         PTRTMP
0283                 01485 Disp7
```

```
0283   27D6          01486                call          Msg
0284   26C1          01487                call          LoadD7         ; LCD position 7
0285   0BB4          01488                decfsz        PTRTMP,F
0286   2A89          01489                goto          Disp8
0287   301F          01490                movlw         .31
0288   00B4          01491                movwf         PTRTMP
0289                 01492 Disp8
0289   27D6          01493                call          Msg
028A   26DC          01494                call          LoadD8         ; LCD position 8
028B   0BB3          01495                decfsz        POINTER,F
028C   2A8F          01496                goto          MessageWait    ; Wait a second before scrolling
028D   301F          01497                movlw         .31
028E   00B3          01498                movwf         POINTER
                     01499
028F                 01500 MessageWait    ; Wait a second before scrolling
028F   1874          01501                btfsc         BUTTON,0       ; any BUTTONs pressed?
0290   0008          01502                return                       ; yes, continue with program
                     01503
0291   1C73          01504                btfss         UPDATE,0
0292   2A8F          01505                goto          MessageWait    ; wait for 1 second timer tick
                     01506
0293   01F3          01507                clrf          UPDATE         ; 1 second passed, clear counter
0294   2A5D          01508                goto          Message        ; redisplay message, scrolled 1 char left
                     01509
                     01510 ;*************************************************************************
                     01511 ; Writes the current time to the LCD display.
                     01512 ;
                     01513 ; Clear the Display, convert SECOND to BCD, write the BCD result to the 2 seconds digits.
                     01514 ; Convert MINUTE to BCD, write the BCD result to the 2 minutes digits.
                     01515 ; Convert HOUR to BCD, write the BCD result to the 2 hours digits.
                     01516 ; Turn on the colons.
                     01517 ;*************************************************************************
                     01518 ; Inputs:  SECOND, MINUTE, HOUR
                     01519 ; Outputs: LCD display
                     01520
0295                 01521 UpdateDisplay  ; Writes Time to the LCD
0295   2612          01522                call          ClrLCD         ; Clear the LCD display
0296   0822          01523 UpdDisp        movf          SECOND,W       ; get the HEX value for seconds
0297   00D1          01524                movwf         TEMPL          ; LSB
0298   01D2          01525                clrf          TEMPH          ; MSB (SECONDs are never > 256)
0299   24DC          01526                call          Bin2BCD16      ; change to BCD
029A   0850          01527                movf          R2,W
029B   00F1          01528                movwf         LCDTEMP2
029C   390F          01529                andlw         0x0f
029D   26DC          01530                call          LoadD8         ; Write seconds to LCD digit 8
029E   0E71          01531                swapf         LCDTEMP2,W
```

```
029F   390F           01532                andlw          0x0f
02A0   26C1           01533                call           LoadD7          ; and 7
02A1   0823           01534                movf           MINUTE,W        ; Display MINUTEs
02A2   00D1           01535                movwf          TEMPL
02A3   01D2           01536                clrf           TEMPH
02A4   24DC           01537                call           Bin2BCD16
02A5   0850           01538                movf           R2,W
02A6   00F1           01539                movwf          LCDTEMP2
02A7   390F           01540                andlw          0x0f
02A8   26A6           01541                call           LoadD6          ; On LCD digit 6
02A9   0E71           01542                swapf          LCDTEMP2,W
02AA   390F           01543                andlw          0x0f
02AB   268B           01544                call           LoadD5          ; and 5
02AC   0824           01545                movf           HOUR,W          ; Display HOURs
02AD   00D1           01546                movwf          TEMPL
02AE   01D2           01547                clrf           TEMPH
02AF   24DC           01548                call           Bin2BCD16
02B0   0850           01549                movf           R2,W
02B1   00F1           01550                movwf          LCDTEMP2
02B2   390F           01551                andlw          0x0f
02B3   2670           01552                call           LoadD4          ; On LCD digit 4
02B4   0E71           01553                swapf          LCDTEMP2,W
02B5   390F           01554                andlw          0x0f
02B6   2655           01555                call           LoadD3          ; and 3
02B7   1283 1703      01556                banksel        LCDD11
02B9   149B           01557                bsf            COLON2          ; Turn on the colons
02BA   141B           01558                bsf            COLON3
02BB   1283 1303      01559                banksel        PORTA
02BD   1073           01560                bcf            UPDATE,0
02BE   0008           01561                return
                      01562
                      01563 ;*****************************************************************************
                      01564 ; Read data from the CS4560 through the SSP PORT                             *
                      01565 ;*****************************************************************************
                      01566 ; Inputs: TXDATA (read command specifying which register to read)
                      01567 ; Outputs: RXDATA0, RXDATA1, RXDATA2 (received data buffer)
                      01568
02BF                  01569 SSPRead
02BF   1105           01570                bcf            CS              ; CS4560 chip select pin7 low
02C0   083C           01571                movf           TXDATA,W        ; CS4560 register to read
02C1   0093           01572                movwf          SSPBUF          ; put it in SSPBUF the start read
02C2   118C           01573                bcf            PIR1,SSPIF      ; clear flag
02C3   1D8C           01574                btfss          PIR1,SSPIF      ; wait for SSP to be ready
02C4   2AC3           01575                goto           $-1
02C5   118C           01576                bcf            PIR1,SSPIF      ; clear flag
02C6   0813           01577                movf           SSPBUF,W
```

```
                    01578
                    01579 ; send three dummy bytes
                    01580 ; in this case we send "SYNC1"s
                    01581
02C7   30FF         01582               movlw       SYNC1          ; dummy byte 1
02C8   0093         01583               movwf       SSPBUF
02C9   118C         01584               bcf         PIR1,SSPIF
02CA   1D8C         01585               btfss       PIR1,SSPIF
02CB   2ACA         01586               goto        $-1
02CC   118C         01587               bcf         PIR1,SSPIF
02CD   0813         01588               movf        SSPBUF,W       ; read the returned data and put in variable
02CE   00B9         01589               movwf       RXDATA0        ; MSB
                    01590
02CF   30FF         01591               movlw       SYNC1          ; dummy byte 2
02D0   0093         01592               movwf       SSPBUF
02D1   118C         01593               bcf         PIR1,SSPIF
02D2   1D8C         01594               btfss       PIR1,SSPIF
02D3   2AD2         01595               goto        $-1
02D4   118C         01596               bcf         PIR1,SSPIF
02D5   0813         01597               movf        SSPBUF,W
02D6   00BA         01598               movwf       RXDATA1
02D7   30FF         01599               movlw       SYNC1          ; dummy byte 3
02D8   0093         01600               movwf       SSPBUF
02D9   118C         01601               bcf         PIR1,SSPIF
02DA   1D8C         01602               btfss       PIR1,SSPIF
02DB   2ADA         01603               goto        $-1
02DC   118C         01604               bcf         PIR1,SSPIF
02DD   0813         01605               movf        SSPBUF,W
02DE   00BB         01606               movwf       RXDATA2        ; LSB
                    01607
02DF   1505         01608               bsf         CS             ; CS4560 chip select pin back high
02E0   0008         01609               return
                    01610
                    01611 ;*****************************************************************************
                    01612 ; Write data to a CS4560 register                                            *
                    01613 ; The commmand will be in TXDATA and the data will be in                     *
                    01614 ; TXDATA0 (MSB), TXDATA1, and TXDATA2 (LSB)                                  *
                    01615 ;*****************************************************************************
                    01616 ; Inputs:  TXDATA, TXDATA1, TXDATA2, TXDATA3
                    01617 ; Outputs: none
                    01618
02E1                01619 SSPWrite
02E1   1105         01620               bcf         CS             ; set CS4560 chip select pin 7 low
02E2   118C         01621               bcf         PIR1,SSPIF     ; clear the flag
                    01622
02E3   083C         01623               movf        TXDATA,W       ; get command (or register)
```

```
02E4   0093          01624              movwf          SSPBUF          ; put it in SSPBUF
02E5   1D8C          01625              btfss          PIR1,SSPIF      ; wait for SSP to be ready
02E6   2AE5          01626              goto           $-1
02E7   0813          01627              movf           SSPBUF,W        ; dummy read, discard byte
02E8   118C          01628              bcf            PIR1,SSPIF      ; command byte sent, clear flag
                     01629
02E9   083D          01630              movf           TXDATA0,W       ; get first data byte
02EA   0093          01631              movwf          SSPBUF          ; send data byte 0 (MSB)
02EB   1D8C          01632              btfss          PIR1,SSPIF      ; wait for SSP to be ready
02EC   2AEB          01633              goto           $-1
02ED   0813          01634              movf           SSPBUF,W        ; dummy read, discard byte
02EE   118C          01635              bcf            PIR1,SSPIF      ; first data byte sent, clear flag
                     01636
02EF   083E          01637              movf           TXDATA1,W
02F0   0093          01638              movwf          SSPBUF          ; send data byte 1
02F1   1D8C          01639              btfss          PIR1,SSPIF      ; wait for SSP to be ready
02F2   2AF1          01640              goto           $-1
02F3   0813          01641              movf           SSPBUF,W
02F4   118C          01642              bcf            PIR1,SSPIF      ; second data byte sent, clear flag
                     01643
02F5   083F          01644              movf           TXDATA2,W
02F6   0093          01645              movwf          SSPBUF          ; send data byte 2 (LSB)
02F7   1D8C          01646              btfss          PIR1,SSPIF      ; wait for SSP to be ready
02F8   2AF7          01647              goto           $-1
02F9   0813          01648              movf           SSPBUF,W        ; dummy read, discard byte
02FA   118C          01649              bcf            PIR1,SSPIF      ; third data byte sent, clear flag
                     01650
02FB   1505          01651              bsf            CS              ; set CS4560 chip select pin back high
                     01652
02FC   0008          01653              return
                     01654
                     01655 ;*************************************************************************
                     01656 ; Write a 1 byte command to the CS4560                                   *
                     01657 ; The command will be in TXDATA                                                  *
                     01658 ;*************************************************************************
                     01659 ; Inputs: TXDATA
                     01660 ; Outputs: none
                     01661
02FD                 01662 SSPCmd
02FD   1105          01663              bcf            CS              ; set CS4560 chip select pin 7 low
                     01664
02FE   083C          01665              movf           TXDATA,W        ; get command
02FF   0093          01666              movwf          SSPBUF          ; send command
0300   118C          01667              bcf            PIR1,SSPIF      ; clear interrupt flag
0301   1D8C          01668              btfss          PIR1,SSPIF      ; loop waiting for command to be sent
0302   2B01          01669              goto           $-1
```

```
0303   0813           01670                movf           SSPBUF,W       ; command sent
                      01671
0304   1505           01672                bsf            CS             ; set CS4560 chip select pin back high
                      01673
0305   0008           01674                return
                      01675
                      01676 ;**************** Calibrate  ***************************
                      01677 ; This routine allows the user to temporarily change the calibration
                      01678 ; values used in the CS4560.
                      01679 ;
                      01680 ; These values are NOT written to the EEPROM and will be lost when the
                      01681 ; power to the unit is lost. Some other method is required to write
                      01682 ; these values to the EEPROM.
                      01683 ;
                      01684 ; This routine is executed by holding BUTTONs 1 and 4 and while pressing MCLR.
                      01685 ;
                      01686 ; Prior to calling this routine CALMODE and CALDIG are set to 0x01
                      01687 ; (voltage offset MODE and Digit 1)
                      01688 ;
                      01689 ; CAL "MODE" is first displayed on the LCD
                      01690 ; CAL EOFF = Voltage Offset,  CAL E GA = Voltage Gain
                      01691 ; CAL COFF = Current Offset,  CAL C GA = Current Gain
                      01692 ; CAL P GA = Pulse Rate gain
                      01693 ;
                      01694 ; That value is read from the CS4560 and displayed on the LCD.
                      01695 ;
                      01696 ; BUTTON 1 writes the current value to the CS4560
                      01697 ;
                      01698 ; BUTTON 2 steps through the Modes (V GA, V OFF, C GA, C OFF, P GA)
                      01699 ;
                      01700 ; Button 3 selects the next hex digit to modify. The decimal point next to
                      01701 ; the selected digit is turned on.
                      01702 ;
                      01703 ; BUTTON 4 increments the digit, when "F" is reached the digit rolls over to
                      01704 ; 0.  Only one digit is changed (overflowing the lower nibble does
                      01705 ; not increment upper nibble).
                      01706 ;
                      01707 ; Reset (!MCLR) exits this routine.
                      01708 ;
                      01709 ; CALMODE (7 6 5 4 3 2 1 0)   selects calibration variable to adjust
                      01710 ; 7 = don't care     6 = don't care      5 = don't care      4 = Pulse rate gain
                      01711 ; 3 = current gain   2 = voltage gain    1 = current offset  0 = voltage offset
                      01712 ;*******************************************************************
                      01713 ; Inputs:  CALMODE, BUTTON, CALDIG
                      01714 ; Outputs:
                      01715
```

```
0306                    01716 Calibrate
0306   01F4             01717                clrf            BUTTON          ; clear button state
                        01718
0307   3006             01719                movlw           b'00000110'     ; read Voltage Offset command
0308   00BC             01720                movwf           TXDATA
0309   22BF             01721                call            SSPRead         ; get voltage offset from CS5460
                        01722
030A   2612             01723                call            ClrLCD          ; blank the LCD display
030B   300C             01724                movlw           0x0C            ; "C"
030C   261F             01725                call            LoadD1
030D   300A             01726                movlw           0x0A            ; "A"
030E   263A             01727                call            LoadD2
030F   301D             01728                movlw           0x1D            ; "L"
0310   2655             01729                call            LoadD3
                        01730
0311   0836             01731                movf            CALMODE,W
0312   3905             01732                andlw           B'00000101'
0313   1903             01733                btfsc           STATUS,Z        ; test for voltage mode
0314   2B17             01734                goto            CalCur          ; not voltage mode
                        01735
0315   300E             01736                movlw           0x0E            ; assume voltage mode, "E"
0316   2B1E             01737                goto            SendE5          ; display "E"
                        01738
0317   0836             01739 CalCur         movf            CALMODE,W
0318   390A             01740                andlw           B'00001010'
0319   1903             01741                btfsc           STATUS,Z        ; test for current mode
031A   2B1D             01742                goto            CalPul          ; not current mode
                        01743
031B   300C             01744                movlw           0x0C            ; assume Pulse Rate mode, "C"
031C   2B1E             01745                goto            SendE5          ; display "C"
                        01746
031D   3010             01747 CalPul         movlw           0x10            ; assume Pulse Rate mode, "P"
                        01748
031E   268B             01749 SendE5         call            LoadD5          ; display "E", "C", or "P"
                        01750
031F   0836             01751                movf            CALMODE,W
0320   3903             01752                andlw           B'00000011'
0321   1D03             01753                btfss           STATUS,Z        ; test for offset mode
0322   2B28             01754                goto            SendOff         ; display "OFF"
                        01755
                        01756                ; display "GA"
0323   3006             01757 SendGA         movlw           0x06            ; "G"
0324   26C1             01758                call            LoadD7
0325   300A             01759                movlw           0x0A            ; "A"
0326   26DC             01760                call            LoadD8
0327   2B2E             01761                goto            Cal1
```

```
                    01762
0328                01763 SendOff       ; display "OFF"
0328  3000           01764               movlw           0x00            ; "O"
0329  26A6           01765               call            LoadD6
032A  300F           01766               movlw           0x0F            ; "F"
032B  26C1           01767               call            LoadD7
032C  300F           01768               movlw           0x0F            ; "F"
032D  26DC           01769               call            LoadD8
                    01770
032E                01771 Cal1
032E  1D74           01772               btfss           BUTTON,2        ; was BUTTON 2 pressed? (step through modes)
032F  2B2E           01773               goto            Cal1            ; no, wait some more
                    01774
0330                01775 DispCal
0330  01F4           01776               clrf            BUTTON          ; clear button state
                    01777
0331  01B7           01778               clrf            CALDIG
0332  1437           01779               bsf             CALDIG,0        ; digit 1 selected
                    01780
0333  3006           01781               movlw           b'00000110'     ; CS5460 command - Read Voltage Offset
0334  1836           01782               btfsc           CALMODE,0       ; test for Voltage Offset MODE
0335  2B40           01783               goto            GetCal          ; read CS5460 register
                    01784
0336  3002           01785               movlw           b'00000010'     ; CS5460 command - Read Current Offset
0337  18B6           01786               btfsc           CALMODE,1       ; test for Current Offset MODE
0338  2B40           01787               goto            GetCal          ; read CS5460 register
                    01788
0339  3008           01789               movlw           b'00001000'     ; CS5460 command - Read Voltage Gain
033A  1936           01790               btfsc           CALMODE,2       ; test for Voltage Gain MODE
033B  2B40           01791               goto            GetCal          ; read CS5460 register
                    01792
033C  3004           01793               movlw           b'00000100'     ; CS5460 command - Read Current Gain
033D  19B6           01794               btfsc           CALMODE,3       ; test for Current Gain MODE
033E  2B40           01795               goto            GetCal          ; read CS5460 register
                    01796
033F  300C           01797               movlw           b'00001100'     ; CS5460 command - Read Pulse Rate Gain
                    01798
0340                01799 GetCal        ; read CS5460 register
0340  00BC           01800               movwf           TXDATA          ; move data to TX command buffer
0341  22BF           01801               call            SSPRead         ; get data from CS5460
0342  26F7           01802               call            WriteLCD        ; send HEX data to display
                    01803
0343  1703           01804               bsf             STATUS,RP1      ; bank 2
0344  159A           01805               bsf             DEC1            ; turn on decimal point next to digit 1
0345  1303           01806               bcf             STATUS,RP1      ; bank 0
                    01807
```

```
0346   01F4           01808                clrf          BUTTON         ; clear all BUTTON readings
0347                  01809 Cal2
0347   1C74           01810                btfss         BUTTON,0       ; was a BUTTON pressed?
0348   2B47           01811                goto          Cal2           ; no, wait some more
                      01812
0349   18F4           01813                btfsc         BUTTON,1       ; was BUTTON 1 pressed? (write to CS5460)
034A   2BC5           01814                goto          WriteCS4560    ; yes, copy RXDATA buffer to TXDATA buffer and send
                      01815
034B   1D74           01816                btfss         BUTTON,2       ; was BUTTON 2 pressed? (step through MODEs)
034C   2B55           01817                goto          ChkButton3     ; check another BUTTON
                      01818
034D                  01819 IncCal
034D   01F4           01820                clrf          BUTTON         ; clear button states
                      01821
034E   1003           01822                bcf           STATUS,C       ; clear carry bit before rotate
                      01823
034F   0DB6           01824                rlf           CALMODE,F      ; step to next calibrate MODE
0350   1EB6           01825                btfss         CALMODE,5      ; test for valid calibrate MODE
0351   2B06           01826                goto          Calibrate      ; back to top of calibrate MODE loop (start over)
                      01827
0352   12B6           01828                bcf           CALMODE,5      ; clear invalid MODE
0353   1436           01829                bsf           CALMODE,0      ; reset to Voltage Offset
0354   2B06           01830                goto          Calibrate      ; back to top of calibrate MODE loop (start over)
                      01831
0355                  01832 ChkButton3
0355   19F4           01833                btfsc         BUTTON,3       ; was BUTTON 3 pressed? (select hex digit)
0356   2B5B           01834                goto          NextDigit      ; yes
0357   1A74           01835                btfsc         BUTTON,4       ; was BUTTON 4 pressed? (increment digit)
0358   2B8C           01836                goto          WhichDigit     ; yes
                      01837
0359   01F4           01838                clrf          BUTTON         ; should not get here
035A   2B47           01839                goto          Cal2           ; wait for a BUTTON press
                      01840
035B                  01841 NextDigit
035B   01F4           01842                clrf          BUTTON         ; clear BUTTON data
035C   1003           01843                bcf           STATUS,C       ; clear carry bit before rotate instruction
035D   0DB7           01844                rlf           CALDIG,F       ; select next higher digit
035E   1F37           01845                btfss         CALDIG,6       ; test for digit7 (invalid digit)
035F   2B62           01846                goto          WhichDec       ; valid digit selected
                      01847
0360   3001           01848                movlw         0x01           ; invalid digit selected
0361   00B7           01849                movwf         CALDIG         ; Reset CALDIG to digit 1 (MSB)
0362                  01850 WhichDec
0362   01F4           01851                clrf          BUTTON         ; clear BUTTON data
                      01852
0363   1837           01853                btfsc         CALDIG,0       ; is digit 1 selected?
```

```
0364  2B6E         goto    DigitOne        ; yes, modify digit 1

0365  18B7         btfsc   CALDIG,1        ; is digit 2 selected?
0366  2B73         goto    DigitTwo        ; yes, modify digit 2

0367  1937         btfsc   CALDIG,2        ; is digit 3 selected?
0368  2B78         goto    DigitThree      ; yes, modify digit 3

0369  19B7         btfsc   CALDIG,3        ; is digit 4 selected?
036A  2B7D         goto    DigitFour       ; yes, modify digit 4

036B  1A37         btfsc   CALDIG,4        ; is digit 5 selected?
036C  2B82         goto    DigitFive       ; yes, modify digit 5

036D  2B87         goto    DigitSix        ; modify digit 6
036E         DigitOne
036E  1703         bsf     STATUS,RP1      ; select bank 2
036F  1218         bcf     DEC7            ; turn off decimal point 7
0370  159A         bsf     DEC1            ; turn on decimal point 1
0371  1303         bcf     STATUS,RP1      ; select bank 0
0372  2B47         goto    Cal2            ; wait for BUTTON press
0373         DigitTwo
0373  1703         bsf     STATUS,6        ; select bank 2
0374  119A         bcf     DEC1            ; turn off decimal point 1
0375  169A         bsf     DEC2            ; turn on decimal point 2
0376  1303         bcf     STATUS,RP1      ; select bank 0
0377  2B47         goto    Cal2            ; wait for BUTTON press
0378         DigitThree
0378  1703         bsf     STATUS,6        ; select bank 2
0379  129A         bcf     DEC2            ; turn off decimal point 2
037A  1799         bsf     DEC3            ; turn on decimal point 3
037B  1303         bcf     STATUS,RP1      ; select bank 0
037C  2B47         goto    Cal2            ; wait for BUTTON press
037D         DigitFour
037D  1703         bsf     STATUS,6        ; select bank 2
037E  1399         bcf     DEC3            ; turn off decimal point 3
037F  1699         bsf     DEC4            ; turn on decimal point 4
0380  1303         bcf     STATUS,RP1      ; select bank 0
0381  2B47         goto    Cal2            ; wait for BUTTON press
0382         DigitFive
0382  1703         bsf     STATUS,6        ; select bank 2
0383  1299         bcf     DEC4            ; turn off decimal point 4
0384  1798         bsf     DEC6            ; turn on decimal point 6
0385  1303         bcf     STATUS,RP1      ; select bank 0
0386  2B47         goto    Cal2            ; wait for BUTTON press
0387         DigitSix
```

```
0387   1703            01900                bsf            STATUS,6       ; select bank 2
0388   1398            01901                bcf            DEC6           ; turn off decimal point 6
0389   1618            01902                bsf            DEC7           ; turn on decimal point 7
038A   1303            01903                bcf            STATUS,RP1     ; select bank 0
038B   2B47            01904                goto           Cal2           ; wait for BUTTON press
                       01905
038C                   01906 WhichDigit
038C   01F4            01907                clrf           BUTTON         ; clear BUTTON data
                       01908
038D   1837            01909                btfsc          CALDIG,0       ; is digit 1 selected?
038E   2B98            01910                goto           IncOne         ; yes, increment digit 1
                       01911
038F   18B7            01912                btfsc          CALDIG,1       ; is digit 2 selected?
0390   2B9D            01913                goto           IncTwo         ; yes, increment digit 2
                       01914
0391   1937            01915                btfsc          CALDIG,2       ; is digit 3 selected?
0392   2BA7            01916                goto           IncThree       ; yes, increment digit 3
                       01917
0393   19B7            01918                btfsc          CALDIG,3       ; is digit 4 selected?
0394   2BAC            01919                goto           IncFour        ; yes, increment digit 4
                       01920
0395   1A37            01921                btfsc          CALDIG,4       ; is digit 5 selected?
0396   2BB6            01922                goto           IncFive        ; yes, increment digit 5
                       01923
0397   2BBB            01924                goto           IncSix         ; increment digit 6
                       01925
0398   0839            01926 IncOne         movf           RXDATA0,W      ; get digit data
0399   3E10            01927                addlw          0x10           ; increment digit
039A   00B9            01928                movwf          RXDATA0        ; store digit data
039B   26F7            01929                call           WriteLCD       ; update display
039C   2B62            01930                goto           WhichDec       ; turn on decimal point 1
                       01931
039D   0839            01932 IncTwo         movf           RXDATA0,W      ; get digit data
039E   3E01            01933                addlw          0x01           ; increment digit
039F   1C83            01934                btfss          STATUS,DC      ; was there a digit overflow?
03A0   2BA4            01935                goto           IncTwo2        ; no, display digit
                       01936
03A1   00B9            01937                movwf          RXDATA0        ; save result to buffer
03A2   3010            01938                movlw          0x10
03A3   0239            01939                subwf          RXDATA0,W      ; decrement upper digit
                       01940
03A4   00B9            01941 IncTwo2        movwf          RXDATA0        ; save result
03A5   26F7            01942                call           WriteLCD       ; update display
03A6   2B62            01943                goto           WhichDec       ; turn on decimal point 2
                       01944
03A7   083A            01945 IncThree       movf           RXDATA1,W      ; get digit data
```

```
03A8   3E10            01946                addlw          0x10
03A9   00BA            01947                movwf          RXDATA1        ; increment  digit
03AA   26F7            01948                call           WriteLCD       ; update display
03AB   2B62            01949                goto           WhichDec       ; turn on decimal point 3
                       01950
03AC   083A            01951 IncFour        movf           RXDATA1,W      ; get digit data
03AD   3E01            01952                addlw          0x01           ; increment digit
03AE   1C83            01953                btfss          STATUS,DC      ; was there a digit overflow?
03AF   2BB3            01954                goto           IncFour2       ; no, display digit
                       01955
03B0   00BA            01956                movwf          RXDATA1        ; save result
03B1   3010            01957                movlw          0x10
03B2   023A            01958                subwf          RXDATA1,W      ; decrement upper digit
                       01959
03B3   00BA            01960 IncFour2       movwf          RXDATA1        ; save result
03B4   26F7            01961                call           WriteLCD       ; update display
03B5   2B62            01962                goto           WhichDec       ; turn on decimal point 4
                       01963
03B6   083B            01964 IncFive        movf           RXDATA2,W      ; get digit data
03B7   3E10            01965                addlw          0x10           ; increment digit
03B8   00BB            01966                movwf          RXDATA2        ; save result
03B9   26F7            01967                call           WriteLCD       ; update display
03BA   2B62            01968                goto           WhichDec       ; turn on decimal point 5
                       01969
03BB   083B            01970 IncSix         movf           RXDATA2,W      ; get digit data
03BC   3E01            01971                addlw          0x01           ; increment digit
03BD   1C83            01972                btfss          STATUS,DC      ; was there a digit overflow?
03BE   2BC2            01973                goto           IncSix2        ; no, display digit
                       01974
03BF   00BB            01975                movwf          RXDATA2        ; save result
03C0   3010            01976                movlw          0x10
03C1   023B            01977                subwf          RXDATA2,W      ; decrement upper digit
                       01978
03C2   00BB            01979 IncSix2        movwf          RXDATA2        ; save result
03C3   26F7            01980                call           WriteLCD       ; update display
03C4   2B62            01981                goto           WhichDec       ; turn on decimal point 6
                       01982
03C5                   01983 WriteCS4560    ; copies RXDATA buffer to TXDATA buffer
03C5   01F4            01984                clrf           BUTTON
03C6   173C            01985                bsf            TXDATA,6       ; set to Write MODE
03C7   0839            01986                movf           RXDATA0,W      ; copy RX buffer (modified) to TX buffer
03C8   00BD            01987                movwf          TXDATA0
03C9   083A            01988                movf           RXDATA1,W
03CA   00BE            01989                movwf          TXDATA1
03CB   083B            01990                movf           RXDATA2,W
03CC   00BF            01991                movwf          TXDATA2
```

```
03CD   22E1          01992                call           SSPWrite       ; transmit TX buffer
03CE   2B4D          01993                goto           IncCal         ; select next calibration MODE
03CF   0008          01994                return
                     01995
03D0                 01996 Delay10mS
                     01997                ; 10ms delay routine
03D0   1283          01998                bcf            STATUS,RP0
03D1   1303          01999                bcf            STATUS,RP1
03D2   01A0          02000                clrf           TEMPA
03D3   3005          02001                movlw          .5
03D4   00A1          02002                movwf          TEMPB
03D5   2BD6          02003                goto           Dly1
                     02004
03D6                 02005 Dly1           ; the delay loop
03D6   2BD7          02006                goto           $+1
03D7   2BD8          02007                goto           $+1
03D8   0000          02008                nop
03D9   0BA0          02009                decfsz         TEMPA,F
03DA   2BD6          02010                goto           Dly1
03DB   0BA1          02011                decfsz         TEMPB,F
03DC   2BD6          02012                goto           Dly1
03DD   0008          02013                return
                     02014
                     02015 ;************************  I2C EEPROM routines  **************************
                     02016 ; These routines write to and read from the 24C01 EEPROM
                     02017 ; SDA has a pull-up resistor, but SCL does not.
                     02018 ; SCL is driven high rather than allowed to be pulled high by pull-up resistor
                     02019 ;*************************************************************************
                     02020
                     02021 ;*************************************************************************
                     02022 ; Sends Start Bit - SDA falls while SCL is high
                     02023 ;*************************************************************************
                     02024 ; Inputs:  none
                     02025 ; Outputs: none
                     02026
03DE                 02027 SendStart      ; start bit - data goes low while clock is high
                     02028
03DE   1485          02029                bsf            SDA            ; PORTA, bit1
03DF   1683          02030                bsf            STATUS,RP0     ; bank 1
03E0   1085          02031                bcf            SDATRIS        ; set SDA as output
03E1   1283          02032                bcf            STATUS,RP0     ; bank 0
03E2   1405          02033                bsf            SCL            ; PORTA, bit0
                     02034
03E3   1085          02035                bcf            SDA            ; data set low
                     02036
03E4   1005          02037                bcf            SCL            ; clock goes low
```

```
                    02038
03E5   0008         02039                return
                    02040
                    02041 ;*****************************************************************************
                    02042 ; Sends Stop Bit - SDA rises while SCL is high
                    02043 ;*****************************************************************************
                    02044 ; Inputs:  none
                    02045 ; Outputs: none
                    02046
03E6                02047 SendStop       ; stop bit - data goes high while clock is high
                    02048
03E6   1085         02049                bcf            SDA            ; data goes low
03E7   1683         02050                bsf            STATUS,RP0     ; bank 1
03E8   1085         02051                bcf            SDATRIS        ; set SDA as output
03E9   1283         02052                bcf            STATUS,RP0     ; bank 0
                    02053
03EA   1405         02054                bsf            SCL            ; clock goes high
                    02055
03EB   1485         02056                bsf            SDA            ; data goes high
03EC   1683         02057                bsf            STATUS,RP0     ; bank 1
03ED   1485         02058                bsf            SDATRIS        ; set SDA as input
03EE   1283         02059                bcf            STATUS,RP0     ; bank 0
                    02060
03EF   0008         02061                return
                    02062
                    02063 ;*****************************************************************************
                    02064 ; BYTECOUNT is loaded with 8 (8 bits to send). For each bit in EETEMP starting
                    02065 ; with MSB, sets SDA to to same state as bit to send, sets SCL high then low.
                    02066 ; BYTECOUNT is decremented for each bit. When BYTECOUNT reaches zero, all bits
                    02067 ; have been sent.
                    02068 ;*****************************************************************************
                    02069 ; Inputs:  EETEMP (data to transmit)
                    02070 ; Outputs: none
                    02071 ; used:    BYTECOUNT (bit counter)
                    02072
03F0                02073 SendData       ; generates clock, reads SDA
                    02074
03F0   3008         02075                movlw          0x08           ; send eight bits
03F1   00D4         02076                movwf          BYTECOUNT
                    02077
03F2   1683         02078                bsf            STATUS,RP0     ; bank 1
03F3   1085         02079                bcf            SDATRIS        ; SDA set to output
03F4   1283         02080                bcf            STATUS,RP0     ; bank 0
                    02081
03F5                02082 SendDataLoop
03F5   1BD7         02083                btfsc          EETEMP,7       ; EETEMP MSB
```

```
03F6   2BF9          02084                goto          SDAHigh
                     02085
03F7   1085          02086 SDALow         bcf           SDA            ; SDA goes low
03F8   2BFA          02087                goto          SCLPulse
                     02088
03F9   1485          02089 SDAHigh        bsf           SDA            ; SDA goes high
                     02090
03FA   0000          02091 SCLPulse       nop
03FB   1405          02092                bsf           SCL            ; clock pulse goes high
03FC   0000          02093                nop
03FD   0000          02094                nop
03FE   1005          02095                bcf           SCL            ; clock pulses goes low
                     02096
03FF   0DD7          02097 DecCntA        rlf           EETEMP,F       ; rotate data to transmit
0400   0BD4          02098                decfsz        BYTECOUNT,F    ; count bit as sent
0401   2BF5          02099                goto          SendDataLoop   ; not done yet, repeat loop
0402   0008          02100                return                       ; done with loop, return
                     02101
                     02102 ;*************************************************************************
                     02103 ; BYTECOUNT is loaded with 8 (8 bits to receive). SCL is set high, SDA is read,
                     02104 ; and SCL is set low. BYTECOUNT is decremented for each bit. The received bit
                     02105 ; is shifted into EETEMP. When BYTECOUNT reaches zero, all bits have been
                     02106 ; received, with the received byte in EETEMP.
                     02107 ;*************************************************************************
                     02108 ; Inputs:  none
                     02109 ; Outputs: EEDATA (received data)
                     02110 ; used:    BYTECOUNT (bit counter)
                     02111
0403   3008          02112 GetData        movlw         0x08
0404   00D4          02113                movwf         BYTECOUNT      ; init bit counter
                     02114
0405   1405          02115 ReadData       bsf           SCL            ; clock high
                     02116
0406   1885          02117                btfsc         SDA            ; read data
0407   2C0A          02118                goto          RDHigh
0408   1056          02119                bcf           EEDATA,0
0409   2C0B          02120                goto          DRCount
                     02121
040A   1456          02122 RDHigh         bsf           EEDATA,0
                     02123
040B   1005          02124 DRCount        bcf           SCL            ; clock low
040C   0BD4          02125                decfsz        BYTECOUNT,F
040D   2C0F          02126                goto          NextBit
040E   0008          02127                return
                     02128
040F   0DD6          02129 NextBit        rlf           EEDATA,F
```

```
0410   2C05          02130                goto          ReadData
                     02131
                     02132 ;*****************************************************************************
                     02133 ; Generates a 9th clock pulsse so slave can send ACK bit. This bit is not
                     02134 ; recorded.
                     02135 ;*****************************************************************************
                     02136 ; Inputs:  none
                     02137 ; Outputs: none
                     02138
0411                 02139 GetACK
0411   1683          02140                bsf           STATUS,RP0     ; bank 1
0412   1485          02141                bsf           SDATRIS        ; set SDA to input
0413   1283          02142                bcf           STATUS,RP0     ; bank 0
                     02143
0414   1405          02144                bsf           SCL            ; clock pulse for ACK
0415   0000          02145                nop
0416   0000          02146                nop
0417   0000          02147                nop
                     02148
0418   1005          02149                bcf           SCL
                     02150
0419   0008          02151                return
                     02152
                     02153 ;*****************************************************************************
                     02154 ; Single byte write to EEPROM.
                     02155 ;
                     02156 ; Sends write command to EEPROM followed by address in EEPROM (EEADDR) and data
                     02157 ; to write (EEDATA).
                     02158 ; The delay allows time for the EEPROM to perform the write operation.
                     02159 ;*****************************************************************************
                     02160 ; Inputs:  EEADDR, EEDATA
                     02161 ; Outputs: none
                     02162
041A   30A0          02163 EEWrite        movlw         b'10100000'    ; Send Write Code to EEPROM memory
041B   00D7          02164                movwf         EETEMP
                     02165
041C   23DE          02166                call          SendStart      ; send Start Bit
041D   23F0          02167                call          SendData       ; send 8 bits of data
041E   2411          02168                call          GetACK         ; get ACK bit from slave
                     02169
041F   0855          02170                movf          EEADDR,W       ; Send memory Address
0420   00D7          02171                movwf         EETEMP
0421   23F0          02172                call          SendData       ; send 8 bits of data
0422   2411          02173                call          GetACK         ; get ACK bit from slave
                     02174
0423   0856          02175                movf          EEDATA,W       ; Send memory Data
```

```
0424   00D7          02176                movwf          EETEMP
0425   23F0          02177                call           SendData       ; send 8 bits of data
0426   2411          02178                call           GetACK         ; get ACK bit from slave
                     02179
0427   23E6          02180                call           SendStop       ; send Stop Bit
                     02181
0428   23D0          02182                call           Delay10mS      ; time for EEPROM write operation
                     02183
0429   0008          02184                return
                     02185
                     02186 ;*************************************************************************
                     02187 ; Single byte read from EEPROM.
                     02188 ;
                     02189 ; Sends read command to EEPROM followed by address in EEPROM (EEADDR). Data
                     02190 ; is received and saved (EEDATA).
                     02191 ;*************************************************************************
                     02192 ; Inputs:  EEADDR
                     02193 ; Outputs: EEDATA
                     02194
042A                 02195 EERead
042A   30A0          02196                movlw          b'10100000'    ; I2C address w/write command
042B   00D7          02197                movwf          EETEMP
                     02198
042C   23DE          02199                call           SendStart      ; send start bit
042D   23F0          02200                call           SendData       ; send I2C address
042E   2411          02201                call           GetACK         ; get ACK bit
                     02202
042F   0855          02203                movf           EEADDR,W       ; memory Address
0430   00D7          02204                movwf          EETEMP
                     02205
0431   23F0          02206                call           SendData       ; send memory address to EEPROM
0432   2411          02207                call           GetACK         ; get ACK bit
                     02208
0433   30A1          02209                movlw          b'10100001'    ; I2C address w/read command
0434   00D7          02210                movwf          EETEMP
                     02211
0435   23DE          02212                call           SendStart      ; send restart bit
0436   23F0          02213                call           SendData       ; send I2C address
0437   2411          02214                call           GetACK         ; get ACK bit
0438   2403          02215                call           GetData        ; get memory data from EEPROM
0439   23E6          02216                call           SendStop       ; send stop bit
                     02217
043A   0008          02218                return
                     02219
                     02220 ;*************************************************************************
                     02221 ; This routine writes the calibration data in the file CAL.INC to the proper
```

```
                    02222 ; address in the EEPROM.
                    02223 ;*****************************************************************************
                    02224 ; Inputs:  contents of CAL.INC (calibration constants)
                    02225 ; Outputs: none
                    02226
043B                02227 WriteSer
043B   3000         02228                movlw          0x00
043C   00D5         02229                movwf          EEADDR         ; select address 0x00
043D   3001         02230                movlw          SERNUMBER      ; write serial number
043E   00D6         02231                movwf          EEDATA
043F   241A         02232                call           EEWrite
                    02233
0440   3001         02234                movlw          0x01
0441   00D5         02235                movwf          EEADDR         ; Voltage Offset MSB, select address 0x01
0442   3010         02236                movlw          VOLTOFFH
0443   00D6         02237                movwf          EEDATA
0444   241A         02238                call           EEWrite
                    02239
0445   3002         02240                movlw          0x02
0446   00D5         02241                movwf          EEADDR         ; Voltage Offset, select address 0x02
0447   301A         02242                movlw          VOLTOFFM
0448   00D6         02243                movwf          EEDATA
0449   241A         02244                call           EEWrite
                    02245
044A   3003         02246                movlw          0x03
044B   00D5         02247                movwf          EEADDR         ; Voltage Offset LSB, select address 0x03
044C   30B6         02248                movlw          VOLTOFFL
044D   00D6         02249                movwf          EEDATA
044E   241A         02250                call           EEWrite
                    02251
044F   3004         02252                movlw          0x04
0450   00D5         02253                movwf          EEADDR         ; Current Offset MSB, select address 0x04
0451   30FE         02254                movlw          CURROFFH
0452   00D6         02255                movwf          EEDATA
0453   241A         02256                call           EEWrite
                    02257
0454   3005         02258                movlw          0x05
0455   00D5         02259                movwf          EEADDR         ; Current Offset, select address 0x05
0456   30DC         02260                movlw          CURROFFM
0457   00D6         02261                movwf          EEDATA
0458   241A         02262                call           EEWrite
                    02263
0459   3006         02264                movlw          0x06
045A   00D5         02265                movwf          EEADDR         ; Current Offset LSB, select address 0x06
045B   3016         02266                movlw          CURROFFL
045C   00D6         02267                movwf          EEDATA
```

```
045D   241A          02268                call          EEWrite
                     02269
045E   3007          02270                movlw         0x07
045F   00D5          02271                movwf         EEADDR         ; Voltage Gain MSB, select address 0x07
0460   3029          02272                movlw         VOLTGAINH
0461   00D6          02273                movwf         EEDATA
0462   241A          02274                call          EEWrite
                     02275
0463   3008          02276                movlw         0x08
0464   00D5          02277                movwf         EEADDR         ; Voltage Gain, select address 0x08
0465   3066          02278                movlw         VOLTGAINM
0466   00D6          02279                movwf         EEDATA
0467   241A          02280                call          EEWrite
                     02281
0468   3009          02282                movlw         0x09
0469   00D5          02283                movwf         EEADDR         ; Voltage Gain LSB, select address 0x09
046A   306F          02284                movlw         VOLTGAINL
046B   00D6          02285                movwf         EEDATA
046C   241A          02286                call          EEWrite
                     02287
046D   300A          02288                movlw         0x0A
046E   00D5          02289                movwf         EEADDR         ; Current Gain MSB, select address 0x0A
046F   302A          02290                movlw         CURRGAINH
0470   00D6          02291                movwf         EEDATA
0471   241A          02292                call          EEWrite
                     02293
0472   300B          02294                movlw         0x0B
0473   00D5          02295                movwf         EEADDR         ; Current Gain, select address 0x0B
0474   302E          02296                movlw         CURRGAINM
0475   00D6          02297                movwf         EEDATA
0476   241A          02298                call          EEWrite
                     02299
0477   300C          02300                movlw         0x0C
0478   00D5          02301                movwf         EEADDR         ; Current Gain LSB, select address 0x0C
0479   300A          02302                movlw         CURRGAINL
047A   00D6          02303                movwf         EEDATA
047B   241A          02304                call          EEWrite
                     02305
047C   300D          02306                movlw         0x0D
047D   00D5          02307                movwf         EEADDR         ; Watt HOUR LSB, select address 0x0D
047E   01D6          02308                clrf          EEDATA
047F   241A          02309                call          EEWrite
                     02310
0480   300E          02311                movlw         0x0E
0481   00D5          02312                movwf         EEADDR         ; Watt HOUR MSB, select address 0x0E
0482   01D6          02313                clrf          EEDATA
```

```
0483   241A            02314                call            EEWrite
                       02315
0484   300F            02316                movlw           0x0F
0485   00D5            02317                movwf           EEADDR          ; Pulse Rate gain MSB, select address 0x0F
0486   3002            02318                movlw           PULSERATEH
0487   00D6            02319                movwf           EEDATA
0488   241A            02320                call            EEWrite
                       02321
0489   3010            02322                movlw           0x10
048A   00D5            02323                movwf           EEADDR          ; Pulse Rate gain, select address 0x10
048B   3014            02324                movlw           PULSERATEM
048C   00D6            02325                movwf           EEDATA
048D   241A            02326                call            EEWrite
                       02327
048E   3011            02328                movlw           0x11
048F   00D5            02329                movwf           EEADDR          ; Pulse Rate gain LSB, select address 0x11
0490   30CB            02330                movlw           PULSERATEL
0491   00D6            02331                movwf           EEDATA
0492   241A            02332                call            EEWrite
                       02333
0493   0008            02334                return
                       02335
                       02336 ;*************************************************************************
                       02337 ; Configure internal peripherals (except LCD) for operation.
                       02338 ; Also configures interrupts.
                       02339 ;
                       02340 ; A 1 second interrupt is configured depending on whether TMR1OSC is defined.
                       02341 ;
                       02342 ; If defined, Timer1 is enabled, and uses the Timer1 Oscillator to generate
                       02343 ; 1 second interrupts. This requires an external 32,768 Hz crystal.
                       02344 ;
                       02345 ; If not defined, CCP1 is configured to used the system clock in
                       02346 ; Compare mode, interrupt only, special event trigger. This generates 1/2 second interrupts, but the
                       02347 ; interrupt rate is divided by 2 in the TMR1ISR (also configured depending
                       02348 ; on the TMR1OSC defintion). This relies on the system clock. In this
                       02349 ; application, the system clock is provided by the CS5460 and is 4.096 MHz.
                       02350 ; If the clock is a different frequency, adjust CCPR1H:CCPR1L.
                       02351 ;*************************************************************************
                       02352 ; Inputs:  TMR1OSC definition
                       02353 ; Outputs: none
                       02354
0494                   02355 InitPeriph
0494   1186            02356                bcf             PORTB,3
                       02357
0495   1683            02358                bsf             STATUS,RP0      ; Bank 1
                       02359
```

```
0496   3007            02360                movlw          b'00000111'
0497   009F            02361                movwf          ADCON1          ; set PORTA all digital
                       02362
0498   30F2            02363                movlw          b'11110010'     ; RA0=EEPROM SCL, RA1=EEPROM SDA, RA2=CS5460 CS,
0499   0085            02364                movwf          TRISA           ; RA3=NC, RA4=NC, RA5=NC
                       02365
049A   30F1            02366                movlw          b'11110001'     ; RB0=!EOUT, RB4=!SW2
049B   0086            02367                movwf          TRISB           ; RB1=!EDIR, RB5=!SW3
                       02368                                               ; RB2=NC,    RB6=!SW4
                       02369                                               ; RB3=NC,    RB7=!SW5
                       02370
049C   3016            02371                movlw          B'00010110'     ; RC0=T1OSC, RC3=SSP SCK
049D   0087            02372                movwf          TRISC           ; RC1=T1OSC, RC4=SSP SDI
                       02373                                               ; RC2=NC,    RC5=SSP SDO
                       02374
049E   30C0            02375                movlw          b'11000000'     ; setup SSP Module - input sampled at end of output,
049F   0094            02376                movwf          SSPSTAT         ; xmit on rising SCLK
                       02377
04A0   1283            02378                bcf            STATUS,RP0      ; bank 0
                       02379
                       02380 ; The CS5460 CPUCLK output is 4.096MHz and provides the Fosc for the controller.
                       02381
                       02382 #IFDEF TMR1OSC ; use this code if 32kHz Timer1OSC is used
                       02383                               ; (comment out #DEFINE TIMER1OSC at the top of the program)
                       02384
                       02385                movlw          0x7f            ; preload Timer1 to interrupt in 1 second
                       02386                movwf          TMR1H
                       02387                movlw          0xFF
                       02388                movwf          TMR1L
                       02389
                       02390                movlw          b'00001111'     ; setup Timer1 - 1 int/sec
                       02391                movwf          T1CON           ; 1:1 prescale, osc enabled, ext clock, async
                       02392 #ELSE
                       02393 ; The following lines initialize Timer1 and CCP1 in Special Event Trigger MODE.
                       02394 ; If the Timer1 oscillator is used, these lines are disabled by the #DEFINE TIMER1OSC
                       02395 ; statement.
04A1   3031            02396                movlw          b'00110001'     ; setup Timer1 - reset by special event trigger
04A2   0090            02397                movwf          T1CON           ; 8:1 prescale, int clock
                       02398
04A3   01FF            02399                clrf           CCPCOUNT        ; clear interrupt counter
                       02400
04A4   30F9            02401                movlw          0xF9
04A5   0096            02402                movwf          CCPR1H          ; 0.5 second compare
04A6   30FF            02403                movlw          0xFF
04A7   0095            02404                movwf          CCPR1L
                       02405
```

```
04A8   300B          02406                movlw          b'00001011'    ; set up CCP1 compare mode,
04A9   0097          02407                movwf          CCP1CON        ; special event trigger
                     02408 #ENDIF
                     02409
04AA   3039          02410                movlw          b'00111001'    ; setup Timer2 -
04AB   0092          02411                movwf          T2CON          ; 4:1 prescale, 8:1 postscale, TMR2 off
                     02412
04AC   3021          02413                movlw          b'00100001'    ; SSP MODE, Fosc/16 -
04AD   0094          02414                movwf          SSPCON         ; SPI master, Fosc/16, SSP enabled, clk idle low
                     02415
04AE   01F3          02416                clrf           UPDATE         ; clear LCD update flag
04AF   01F4          02417                clrf           BUTTON         ; clear pressed BUTTON status
04B0   01F2          02418                clrf           MODEINC        ; clear display MODE counter
                     02419
04B1   1505          02420                bsf            CS             ; Set CS5460 !CS
                     02421
04B2   301F          02422                movlw          .31            ; set message POINTER to start
04B3   00B3          02423                movwf          POINTER        ; scrolling message
                     02424
04B4   0008          02425                return
                     02426
                     02427 ;*************************************************************************
                     02428 ; This routine allows the user to set HOURs and MINUTEs for the real time clock
                     02429 ; display. (Seconds can not be set).
                     02430 ;*************************************************************************
                     02431 ; Inputs:  BUTTON
                     02432 ; Outputs: SECOND, MINUTE, HOUR
                     02433
04B5                 02434 SetClock
04B5   01F3          02435                clrf           UPDATE
                     02436
04B6   300C          02437                movlw          0x0C           ; "C"
04B7   261F          02438                call           LoadD1
04B8   301D          02439                movlw          0x1D           ; "L"
04B9   263A          02440                call           LoadD2
                     02441
04BA   1873          02442 SetLoop        btfsc          UPDATE,0
04BB   2296          02443                call           UpdDisp        ; update time display
                     02444
04BC   1C74          02445                btfss          BUTTON,0       ; wait for BUTTONs to be pressed
04BD   2CBA          02446                goto           SetLoop
04BE   1CF4          02447                btfss          BUTTON,1       ; if BUTTON 1 then we are done
04BF   2CC1          02448                goto           Chk2Clk
                     02449
04C0   28C9          02450                goto           Continue4      ; and return start the main program
04C1                 02451 Chk2Clk
```

```
04C1   1D74          02452                btfss          BUTTON,2       ; is BUTTON 2 pressed?
04C2   2CC5          02453                goto           Chk3Clk        ; no, check BUTTON 3
04C3   1A74          02454                btfsc          BUTTON,4       ; yes, is BUTTON 4 pressed?
04C4   2CD3          02455                goto           IncHr          ; yes, adjust hours
04C5                 02456 Chk3Clk
04C5   1DF4          02457                btfss          BUTTON,3       ; is BUTTON 3 pressed?
04C6   2CC9          02458                goto           EndButtonClk   ; no
04C7   1A74          02459                btfsc          BUTTON,4       ; yes, is BUTTON 4 pressed?
04C8   2CCB          02460                goto           IncMin         ; yes, adjust minutes
04C9                 02461 EndButtonClk
04C9   01F4          02462                clrf           BUTTON
04CA   2CBA          02463                goto           SetLoop        ; wait for BUTTON press
                     02464
04CB                 02465 IncMin
04CB   01F4          02466                clrf           BUTTON
04CC   0AA3          02467                incf           MINUTE,F       ; increment MINUTEs
04CD   303C          02468                movlw          .60            ; if they roll over 60, reset to 0
04CE   0223          02469                subwf          MINUTE,W
04CF   1903          02470                btfsc          STATUS,Z
04D0   01A3          02471                clrf           MINUTE
04D1   1473          02472                bsf            UPDATE,0       ; force display update
04D2   2CBA          02473                goto           SetLoop
                     02474
04D3                 02475 IncHr
04D3   01F4          02476                clrf           BUTTON
04D4   0AA4          02477                incf           HOUR,F         ; increment HOURs
04D5   3019          02478                movlw          .25            ; if they roll over 24, reset to 0
04D6   0224          02479                subwf          HOUR,W
04D7   1903          02480                btfsc          STATUS,Z
04D8   01A4          02481                clrf           HOUR
04D9   1473          02482                bsf            UPDATE,0       ; force display update
04DA   2CBA          02483                goto           SetLoop
                     02484
04DB   0008          02485                return                        ; return to calling routine in page 0
                     02486
                     02487 ;*****************************************************************************
                     02488 ; Bin2BCD16 - Converts a 16-bit binary number in TEMPH:TEMPL into a 3 byte packed
                     02489 ; BCD number in R0:R1:R2.
                     02490 ; R0 holds the 10 thousands digit
                     02491 ; R1 holds the hundreds and thousands digits
                     02492 ; R2 holds the ones and tens digits
                     02493 ;*****************************************************************************
                     02494 ; Inputs:  TEMPH, TEMPL (16-bit binary number)
                     02495 ; Outputs: R0, R1, R2 (5 digit BCD number)
                     02496 ; Used:    COUNT
                     02497
```

```
04DC                02498 Bin2BCD16
                    02499                 ; Initialize variables
04DC   1003         02500                 bcf             STATUS,C        ; clear carry bit
04DD   3010         02501                 movlw           D'16'
04DE   00D3         02502                 movwf           COUNT           ; init bit counter
04DF   01CE         02503                 clrf            R0              ; clear output
04E0   01CF         02504                 clrf            R1              ; clear output
04E1   01D0         02505                 clrf            R2              ; clear output
04E2                02506 Loop16a2
04E2   0DD1         02507                 rlf             TEMPL,F         ; mult by 2, shift MSb to TEMPH
04E3   0DD2         02508                 rlf             TEMPH,F         ; mult by 2, shift MSb to R2
04E4   0DD0         02509                 rlf             R2,F            ; mult by 2, shift MSb to R1
04E5   0DCF         02510                 rlf             R1,F            ; mult by 2, shift MSb to R0
04E6   0DCE         02511                 rlf             R0,F            ; mult by 2, shift MSb to Carry
04E7   0BD3         02512                 decfsz          COUNT,F         ; decrement bit counter
04E8   2CEA         02513                 goto            AdjDec2
04E9   0008         02514                 return
04EA                02515 AdjDec2
04EA   3050         02516                 movlw           R2
04EB   0084         02517                 movwf           FSR             ; point to R2
04EC   24F4         02518                 call            AdjBCD2
                    02519
04ED   304F         02520                 movlw           R1
04EE   0084         02521                 movwf           FSR             ; point to R1
04EF   24F4         02522                 call            AdjBCD2
                    02523
04F0   304E         02524                 movlw           R0
04F1   0084         02525                 movwf           FSR             ; point to R0
04F2   24F4         02526                 call            AdjBCD2
                    02527
04F3   2CE2         02528                 goto            Loop16a2
                    02529
04F4                02530 AdjBCD2
04F4   3003         02531                 movlw           3
04F5   0700         02532                 addwf           INDF,W          ; W = 3 + Rn
04F6   00C9         02533                 movwf           TEMP            ; TEMP = 3+Rn
04F7   19C9         02534                 btfsc           TEMP,3          ;
04F8   0080         02535                 movwf           INDF            ; Rn=TEMP
04F9   3030         02536                 movlw           30              ; decimal adjust?
04FA   0700         02537                 addwf           INDF,W          ; W=30+Rn
04FB   00C9         02538                 movwf           TEMP            ; TEMP=30+Rn
04FC   1BC9         02539                 btfsc           TEMP,7
04FD   0080         02540                 movwf           INDF            ; Rn=TEMP
                    02541
04FE   0008         02542                 return
                    02543
```

```
                    02544 ;*****************************************************************************
                    02545 ; Interrogate CS5460 for Vrms and Irms
                    02546 ; Calculate Apparent Power (Vrms * Irms = AP)
                    02547 ; store binary result in APH:APL (16-bit number)
                    02548 ;*****************************************************************************
                    02549 ; Inputs:  none
                    02550 ; Outputs: APH, APL
                    02551
04FF                02552 CalcAP
04FF   3018         02553               movlw       b'00011000'             ; read RMS Voltage from CS4560
0500   00BC         02554               movwf       TXDATA
0501   22BF         02555               call        SSPRead
0502   0839         02556               movf        RXDATA0,W               ; get V high byte
0503   00C0         02557               movwf       AARGB0                  ; save
0504   083A         02558               movf        RXDATA1,W               ; get V middle byte
0505   00C1         02559               movwf       AARGB1                  ; save
                    02560
0506   3016         02561               movlw       b'00010110'             ; read RMS Current from the CS4560
0507   00BC         02562               movwf       TXDATA
0508   22BF         02563               call        SSPRead
                    02564
0509   0839         02565               movf        RXDATA0,W               ; get I high byte
050A   00C5         02566               movwf       BARGB0                  ; save
                    02567
050B   083A         02568               movf        RXDATA1,W               ; get I middle byte
050C   00C6         02569               movwf       BARGB1                  ; save
                    02570
050D   25D1         02571               call        FXM1616U                ; multiply I and V to get Apparent Power (fraction of
full scale)
                    02572                                                   ; result in AARGB0 (MSB) to AARGB3 (LSB) (use AARGB0
to AARGB2)
                    02573
050E   3098         02574               movlw       MAXPWRH                 ; full scale power
050F   00C5         02575               movwf       BARGB0
0510   30CE         02576               movlw       MAXPWRL
0511   00C6         02577               movwf       BARGB1
                    02578
0512   25D1         02579               call        FXM1616U                ; multiply AP(fraction) and full scale Power to get
Apparent Power
                    02580                                                   ; result in AARGB0 (MSB) to AARGB3 (LSB)
                    02581
0513   0840         02582               movf        AARGB0,W                ; save Apparent Power high byte
0514   00AD         02583               movwf       APH                     ; APx is VoltAmps in binary
                    02584
0515   0841         02585               movf        AARGB1,W                ; save Apparent Power low byte
0516   00AE         02586               movwf       APL
```

```
                    02587
0517   0008         02588                return
                    02589
                    02590 ;*****************************************************************************
                    02591 ; Get energy over the last second, multiply by 10
                    02592 ; (pulses are 1 pulse / 10Watt*second) and save result in TPH:TPL.
                    02593 ; Result is True Power as Watts in binary.
                    02594 ;*****************************************************************************
                    02595 ; Inputs:  none
                    02596 ; Outputs: TPH, TPL
                    02597
0518                02598 CalcTP
0518   082B         02599                movf           PULDISPH,W                  ; Pulse per second MSB
0519   00C0         02600                movwf          AARGB0
051A   082C         02601                movf           PULDISPL,W                  ; Pulse per second LSB
051B   00C1         02602                movwf          AARGB1
                    02603
051C   01C5         02604                clrf           BARGB0                      ; multiply pulses by 10
051D   300A         02605                movlw          0x0A                        ; to get True Power
051E   00C6         02606                movwf          BARGB1
                    02607
051F   25D1         02608                call           FXM1616U                    ; Math 16bit * 16bit multiply routine
                    02609
0520   0842         02610                movf           AARGB2,W                    ; save results in TPH:TPL as Watts in binary
0521   00AF         02611                movwf          TPH
0522   0843         02612                movf           AARGB3,W
0523   00B0         02613                movwf          TPL
                    02614
0524   0008         02615                return
                    02616
                    02617 ;*****************************************************************************
                    02618 ; The following MATH routines were copied from the Microchip
                    02619 ; MATH library AN617
                    02620 ;*****************************************************************************
                    02621 ;
                    02622 ; 24/16 Bit Unsigned Fixed Point Divide 24/16 -> 24.16
                    02623 ;
                    02624 ; Input:  24 bit unsigned fixed point dividend in AARGB0, AARGB1,AARGB2
                    02625 ;         16 bit unsigned fixed point divisor in BARGB0, BARGB1
                    02626 ;
                    02627 ; Use:    CALL    FXD2416U
                    02628 ;
                    02629 ; Output: 24 bit unsigned fixed point quotient in AARGB0, AARGB1,AARGB2
                    02630 ;         16 bit unsigned fixed point remainder in REMB0, REMB1
                    02631 ;
                    02632 ; Result: AARG, REM  <--  AARG / BARG
```

```
                    02633
0525                02634 FXD2416U
0525  01C7          02635              CLRF        REMB0
0526  01C8          02636              CLRF        REMB1
                    02637
0527  01C9          02638              CLRF        TEMP
                    02639
0528  0D40          02640              RLF         AARGB0,W
0529  0DC8          02641              RLF         REMB1,F
052A  0846          02642              MOVF        BARGB1,W
052B  02C8          02643              SUBWF       REMB1,F
052C  0845          02644              MOVF        BARGB0,W
052D  1C03          02645              BTFSS       STATUS,C
052E  0F45          02646              INCFSZ      BARGB0,W
052F  02C7          02647              SUBWF       REMB0,F
                    02648
0530  0103          02649              CLRW
0531  1C03          02650              BTFSS       STATUS,C
0532  3001          02651              MOVLW       1
0533  02C9          02652              SUBWF       TEMP,F
0534  0DC0          02653              RLF         AARGB0,F
                    02654
0535  3007          02655              MOVLW       7
0536  00CD          02656              MOVWF       LOOPCOUNT
                    02657
0537  0D40          02658 LOOPU2416A   RLF         AARGB0,W
0538  0DC8          02659              RLF         REMB1,F
0539  0DC7          02660              RLF         REMB0,F
053A  0DC9          02661              RLF         TEMP,F
053B  0846          02662              MOVF        BARGB1,W
053C  1C40          02663              BTFSS       AARGB0,LSB
053D  2D48          02664              GOTO        UADD46LA
                    02665
053E  02C8          02666              SUBWF       REMB1,F
053F  0845          02667              MOVF        BARGB0,W
0540  1C03          02668              BTFSS       STATUS,C
0541  0F45          02669              INCFSZ      BARGB0,W
0542  02C7          02670              SUBWF       REMB0,F
0543  0103          02671              CLRW
0544  1C03          02672              BTFSS       STATUS,C
0545  3001          02673              MOVLW       1
0546  02C9          02674              SUBWF       TEMP,F
0547  2D51          02675              GOTO        UOK46LA
                    02676
0548  07C8          02677 UADD46LA     ADDWF       REMB1,F
0549  0845          02678              MOVF        BARGB0,W
```

```
054A    1803           02679                  BTFSC          STATUS,C
054B    0F45           02680                  INCFSZ         BARGB0,W
054C    07C7           02681                  ADDWF          REMB0,F
054D    0103           02682                  CLRW
054E    1803           02683                  BTFSC          STATUS,C
054F    3001           02684                  MOVLW          1
0550    07C9           02685                  ADDWF          TEMP,F
                       02686
0551    0DC0           02687 UOK46LA          RLF            AARGB0,F
                       02688
0552    0BCD           02689                  DECFSZ         LOOPCOUNT,F
0553    2D37           02690                  GOTO           LOOPU2416A
                       02691
0554    0D41           02692                  RLF            AARGB1,W
0555    0DC8           02693                  RLF            REMB1,F
0556    0DC7           02694                  RLF            REMB0,F
0557    0DC9           02695                  RLF            TEMP,F
0558    0846           02696                  MOVF           BARGB1,W
0559    1C40           02697                  BTFSS          AARGB0,LSB
055A    2D65           02698                  GOTO           UADD46L8
                       02699
055B    02C8           02700                  SUBWF          REMB1,F
055C    0845           02701                  MOVF           BARGB0,W
055D    1C03           02702                  BTFSS          STATUS,C
055E    0F45           02703                  INCFSZ         BARGB0,W
055F    02C7           02704                  SUBWF          REMB0,F
0560    0103           02705                  CLRW
0561    1C03           02706                  BTFSS          STATUS,C
0562    3001           02707                  MOVLW          1
0563    02C9           02708                  SUBWF          TEMP,F
0564    2D6E           02709                  GOTO           UOK46L8
                       02710
0565    07C8           02711 UADD46L8         ADDWF          REMB1,F
0566    0845           02712                  MOVF           BARGB0,W
0567    1803           02713                  BTFSC          STATUS,C
0568    0F45           02714                  INCFSZ         BARGB0,W
0569    07C7           02715                  ADDWF          REMB0,F
056A    0103           02716                  CLRW
056B    1803           02717                  BTFSC          STATUS,C
056C    3001           02718                  MOVLW          1
056D    07C9           02719                  ADDWF          TEMP,F
                       02720
056E    0DC1           02721 UOK46L8          RLF            AARGB1,F
                       02722
056F    3007           02723                  MOVLW          7
0570    00CD           02724                  MOVWF          LOOPCOUNT
```

```
                    02725
0571   0D41         02726 LOOPU2416B      RLF             AARGB1,W
0572   0DC8         02727                 RLF             REMB1,F
0573   0DC7         02728                 RLF             REMB0,F
0574   0DC9         02729                 RLF             TEMP,F
0575   0846         02730                 MOVF            BARGB1,W
0576   1C41         02731                 BTFSS           AARGB1,LSB
0577   2D82         02732                 GOTO            UADD46LB
                    02733
0578   02C8         02734                 SUBWF           REMB1,F
0579   0845         02735                 MOVF            BARGB0,W
057A   1C03         02736                 BTFSS           STATUS,C
057B   0F45         02737                 INCFSZ          BARGB0,W
057C   02C7         02738                 SUBWF           REMB0,F
057D   0103         02739                 CLRW
057E   1C03         02740                 BTFSS           STATUS,C
057F   3001         02741                 MOVLW           1
0580   02C9         02742                 SUBWF           TEMP,F
0581   2D8B         02743                 GOTO            UOK46LB
                    02744
0582   07C8         02745 UADD46LB        ADDWF           REMB1,F
0583   0845         02746                 MOVF            BARGB0,W
0584   1803         02747                 BTFSC           STATUS,C
0585   0F45         02748                 INCFSZ          BARGB0,W
0586   07C7         02749                 ADDWF           REMB0,F
0587   0103         02750                 CLRW
0588   1803         02751                 BTFSC           STATUS,C
0589   3001         02752                 MOVLW           1
058A   07C9         02753                 ADDWF           TEMP,F
                    02754
058B   0DC1         02755 UOK46LB         RLF             AARGB1,F
                    02756
058C   0BCD         02757                 DECFSZ          LOOPCOUNT,F
058D   2D71         02758                 GOTO            LOOPU2416B
                    02759
058E   0D42         02760                 RLF             AARGB2,W
058F   0DC8         02761                 RLF             REMB1,F
0590   0DC7         02762                 RLF             REMB0,F
0591   0DC9         02763                 RLF             TEMP,F
0592   0846         02764                 MOVF            BARGB1,W
0593   1C41         02765                 BTFSS           AARGB1,LSB
0594   2D9F         02766                 GOTO            UADD46L16
                    02767
0595   02C8         02768                 SUBWF           REMB1,F
0596   0845         02769                 MOVF            BARGB0,W
0597   1C03         02770                 BTFSS           STATUS,C
```

```
0598   0F45         02771                   INCFSZ      BARGB0,W
0599   02C7         02772                   SUBWF       REMB0,F
059A   0103         02773                   CLRW
059B   1C03         02774                   BTFSS       STATUS,C
059C   3001         02775                   MOVLW       1
059D   02C9         02776                   SUBWF       TEMP,F
059E   2DA8         02777                   GOTO        UOK46L16
                    02778
059F   07C8         02779 UADD46L16         ADDWF       REMB1,F
05A0   0845         02780                   MOVF        BARGB0,W
05A1   1803         02781                   BTFSC       STATUS,C
05A2   0F45         02782                   INCFSZ      BARGB0,W
05A3   07C7         02783                   ADDWF       REMB0,F
05A4   0103         02784                   CLRW
05A5   1803         02785                   BTFSC       STATUS,C
05A6   3001         02786                   MOVLW       1
05A7   07C9         02787                   ADDWF       TEMP,F
                    02788
05A8   0DC2         02789 UOK46L16          RLF         AARGB2,F
                    02790
05A9   3007         02791                   MOVLW       7
05AA   00CD         02792                   MOVWF       LOOPCOUNT
                    02793
05AB   0D42         02794 LOOPU2416C        RLF         AARGB2,W
05AC   0DC8         02795                   RLF         REMB1,F
05AD   0DC7         02796                   RLF         REMB0,F
05AE   0DC9         02797                   RLF         TEMP,F
05AF   0846         02798                   MOVF        BARGB1,W
05B0   1C42         02799                   BTFSS       AARGB2,LSB
05B1   2DBC         02800                   GOTO        UADD46LC
                    02801
05B2   02C8         02802                   SUBWF       REMB1,F
05B3   0845         02803                   MOVF        BARGB0,W
05B4   1C03         02804                   BTFSS       STATUS,C
05B5   0F45         02805                   INCFSZ      BARGB0,W
05B6   02C7         02806                   SUBWF       REMB0,F
05B7   0103         02807                   CLRW
05B8   1C03         02808                   BTFSS       STATUS,C
05B9   3001         02809                   MOVLW       1
05BA   02C9         02810                   SUBWF       TEMP,F
05BB   2DC5         02811                   GOTO        UOK46LC
                    02812
05BC   07C8         02813 UADD46LC          ADDWF       REMB1,F
05BD   0845         02814                   MOVF        BARGB0,W
05BE   1803         02815                   BTFSC       STATUS,C
05BF   0F45         02816                   INCFSZ      BARGB0,W
```

```
05C0   07C7          02817                ADDWF          REMB0,F
05C1   0103          02818                CLRW
05C2   1803          02819                BTFSC          STATUS,C
05C3   3001          02820                MOVLW          1
05C4   07C9          02821                ADDWF          TEMP,F
                     02822
05C5   0DC2          02823 UOK46LC        RLF            AARGB2,F
                     02824
05C6   0BCD          02825                DECFSZ         LOOPCOUNT,F
05C7   2DAB          02826                GOTO           LOOPU2416C
                     02827
05C8   1842          02828                BTFSC          AARGB2,LSB
05C9   2DD0          02829                GOTO           UOK46L
05CA   0846          02830                MOVF           BARGB1,W
05CB   07C8          02831                ADDWF          REMB1,F
05CC   0845          02832                MOVF           BARGB0,W
05CD   1803          02833                BTFSC          STATUS,C
05CE   0F45          02834                INCFSZ         BARGB0,W
05CF   07C7          02835                ADDWF          REMB0,F
                     02836
05D0   3400          02837 UOK46L         RETLW          0x00
                     02838
                     02839 ;*****************************************************************************************
                     02840 ;
                     02841 ;       16x16 Bit Unsigned Fixed Point Multiply 16x16 -> 32
                     02842 ;
                     02843 ;       Input:  16 bit unsigned fixed point multiplicand in AARGB0:AARGB1
                     02844 ;               16 bit unsigned fixed point multiplier in BARGB0:BARGB1
                     02845 ;
                     02846 ;       Use:    CALL    FXM1616U
                     02847 ;
                     02848 ;       Output: 32 bit unsigned fixed point product in AARGB0:AARGB1:AARGB2:AARGB3
                     02849 ;
                     02850 ;       Result: AARG  <--  AARG x BARG
                     02851
05D1                 02852 FXM1616U
05D1   01C2          02853                CLRF           AARGB2         ; clear partial product
05D2   01C3          02854                CLRF           AARGB3
05D3   0840          02855                MOVF           AARGB0,W
05D4   00CA          02856                MOVWF          TEMPB0
05D5   0841          02857                MOVF           AARGB1,W
05D6   00CB          02858                MOVWF          TEMPB1
                     02859
05D7   3008          02860                MOVLW          0x08
05D8   00CD          02861                MOVWF          LOOPCOUNT
                     02862
```

```
05D9                    02863 LOOPUM1616A
05D9   0CC6             02864                 RRF             BARGB1, F
05DA   1803             02865                 BTFSC           STATUS,C
05DB   2DE9             02866                 GOTO            ALUM1616NAP
05DC   0BCD             02867                 DECFSZ          LOOPCOUNT, F
05DD   2DD9             02868                 GOTO            LOOPUM1616A
                        02869
05DE   00CD             02870                 MOVWF           LOOPCOUNT
                        02871
05DF                    02872 LOOPUM1616B
05DF   0CC5             02873                 RRF             BARGB0, F
05E0   1803             02874                 BTFSC           STATUS,C
05E1   2DE7             02875                 GOTO            BLUM1616NAP
05E2   0BCD             02876                 DECFSZ          LOOPCOUNT, F
05E3   2DDF             02877                 GOTO            LOOPUM1616B
                        02878
05E4   01C0             02879                 CLRF            AARGB0
05E5   01C1             02880                 CLRF            AARGB1
05E6   3400             02881                 RETLW           0x00
                        02882
05E7                    02883 BLUM1616NAP
05E7   1003             02884                 BCF             STATUS,C
05E8   2E04             02885                 GOTO            BLUM1616NA
                        02886
05E9                    02887 ALUM1616NAP
05E9   1003             02888                 BCF             STATUS,C
05EA   2DF4             02889                 GOTO            ALUM1616NA
                        02890
05EB                    02891 ALOOPUM1616
05EB   0CC6             02892                 RRF             BARGB1, F
05EC   1C03             02893                 BTFSS           STATUS,C
05ED   2DF4             02894                 GOTO            ALUM1616NA
05EE   084B             02895                 MOVF            TEMPB1,W
05EF   07C1             02896                 ADDWF           AARGB1, F
05F0   084A             02897                 MOVF            TEMPB0,W
05F1   1803             02898                 BTFSC           STATUS,C
05F2   0F4A             02899                 INCFSZ          TEMPB0,W
05F3   07C0             02900                 ADDWF           AARGB0, F
                        02901
05F4                    02902 ALUM1616NA
05F4   0CC0             02903                 RRF             AARGB0, F
05F5   0CC1             02904                 RRF             AARGB1, F
05F6   0CC2             02905                 RRF             AARGB2, F
05F7   0BCD             02906                 DECFSZ          LOOPCOUNT, F
05F8   2DEB             02907                 GOTO            ALOOPUM1616
                        02908
```

```
05F9   3008            02909                MOVLW          0x08
05FA   00CD            02910                MOVWF          LOOPCOUNT
                       02911
05FB                   02912 BLOOPUM1616
05FB   0CC5            02913                RRF            BARGB0, F
05FC   1C03            02914                BTFSS          STATUS,C
05FD   2E04            02915                GOTO           BLUM1616NA
05FE   084B            02916                MOVF           TEMPB1,W
05FF   07C1            02917                ADDWF          AARGB1, F
0600   084A            02918                MOVF           TEMPB0,W
0601   1803            02919                BTFSC          STATUS,C
0602   0F4A            02920                INCFSZ         TEMPB0,W
0603   07C0            02921                ADDWF          AARGB0, F
                       02922
0604                   02923 BLUM1616NA
0604   0CC0            02924                RRF            AARGB0, F
0605   0CC1            02925                RRF            AARGB1, F
0606   0CC2            02926                RRF            AARGB2, F
0607   0CC3            02927                RRF            AARGB3, F
0608   0BCD            02928                DECFSZ         LOOPCOUNT, F
0609   2DFB            02929                GOTO           BLOOPUM1616
                       02930
060A   3400            02931                RETLW          0x00
                       02932
                       02933 ;*****************************************************************************
                       02934 ; This routine contains the basics for controlling the LCD
                       02935 ;               movlw          6               (value to be displayed)
                       02936 ;               call           LoadD3          (LCD digit to display it on)
                       02937 ;                                              (this example 3)
                       02938 ;
                       02939 ; Written by Stan D'Souza  4/12/98.  For presentations using the
                       02940 ; PICDEM3 board with asm firmware
                       02941
                       02942 ;*****************************************************************************
                       02943 ; Initializes the LCD module to drive the LCD
                       02944 ;*****************************************************************************
                       02945 ; Inputs: none
                       02946 ; Ouputs: none
                       02947
060B                   02948 InitLCD
060B   1703            02949                bsf            STATUS,RP1      ; Select bank 2
060C   309E            02950                movlw          b'10011110'     ; operates in sleep, 1/3 mux, 1/3 bias, internal RC osc,
060D   008F            02951                movwf          LCDCON          ; init lcd control register, internal voltage generator used
060E   3002            02952                movlw          0x02            ; 1/3 mux, frame freq = 20kHz/(96*(2+1)) = about 70 Hz
060F   008E            02953                movwf          LCDPS
0610   1303            02954                bcf            STATUS,RP1      ; Select bank 0
```

```
0611   0008            02955                return
                       02956
                       02957 ;*****************************************************************************
                       02958 ; Clears all LCD pixels (blanks the display)
                       02959 ;*****************************************************************************
                       02960 ; Inputs: none
                       02961 ; Ouputs: none
                       02962
0612                   02963 ClrLCD
0612   1703            02964                bsf             STATUS,RP1      ; Select bank 2
0613   0190            02965                clrf            LCDD00          ; clear all LCD ram locations
0614   0191            02966                clrf            LCDD01
0615   0192            02967                clrf            LCDD02
0616   0194            02968                clrf            LCDD04
0617   0195            02969                clrf            LCDD05
0618   0196            02970                clrf            LCDD06
0619   0198            02971                clrf            LCDD08
061A   0199            02972                clrf            LCDD09
061B   019A            02973                clrf            LCDD10
061C   019B            02974                clrf            LCDD11
061D   1303            02975                bcf             STATUS,RP1      ; Select bank 0
061E   0008            02976                return
                       02977
                       02978 ;*****************************************************************************
                       02979 ; Each "LoadDx" accepts data in the W register, saves it in LCDTEMP1,
                       02980 ; calls Get7SegDat table, and displays the segment data returned on the LCD
                       02981 ; display.
                       02982 ;*****************************************************************************
                       02983 ; Inputs: W register
                       02984 ; Ouputs: none
                       02985 ; Uses:   LCDTEMP1
                       02986
061F                   02987 LoadD1         ; Write to LCD digit 1 (left side)
061F   00F0            02988                movwf           LCDTEMP1
0620   27B0            02989                call            Get7SegDat      ; get seven segment data in w
0621   00F0            02990                movwf           LCDTEMP1        ; save in temp
0622   1703            02991                bsf             STATUS,RP1      ; Select bank 2
0623   1112            02992                bcf             D1A
0624   1192            02993                bcf             D1B
0625   1196            02994                bcf             D1C
0626   111A            02995                bcf             D1D
0627   1216            02996                bcf             D1E
0628   1212            02997                bcf             D1F
0629   1116            02998                bcf             D1G
062A   1870            02999                btfsc           LCDTEMP1,0      ; if not set, skip segment
062B   1512            03000                bsf             D1A
```

```
062C   18F0          03001               btfsc          LCDTEMP1,1
062D   1592          03002               bsf            D1B
062E   1970          03003               btfsc          LCDTEMP1,2
062F   1596          03004               bsf            D1C
0630   19F0          03005               btfsc          LCDTEMP1,3
0631   151A          03006               bsf            D1D
0632   1A70          03007               btfsc          LCDTEMP1,4
0633   1616          03008               bsf            D1E
0634   1AF0          03009               btfsc          LCDTEMP1,5
0635   1612          03010               bsf            D1F
0636   1B70          03011               btfsc          LCDTEMP1,6
0637   1516          03012               bsf            D1G
0638   1303          03013               bcf            STATUS,RP1     ; Select bank 0
0639   0008          03014               return
                     03015
063A                 03016 LoadD2        ; Write to LCD digit 2
063A   00F0          03017               movwf          LCDTEMP1
063B   27B0          03018               call           Get7SegDat     ; get seven segment data in w
063C   00F0          03019               movwf          LCDTEMP1       ; save in temp
063D   1703          03020               bsf            STATUS,RP1     ; Select bank 2
063E   1092          03021               bcf            D2A
063F   1292          03022               bcf            D2B
0640   1296          03023               bcf            D2C
0641   109A          03024               bcf            D2D
0642   1316          03025               bcf            D2E
0643   1312          03026               bcf            D2F
0644   1096          03027               bcf            D2G
0645   1870          03028               btfsc          LCDTEMP1,0     ; if not set, skip segment
0646   1492          03029               bsf            D2A
0647   18F0          03030               btfsc          LCDTEMP1,1
0648   1692          03031               bsf            D2B
0649   1970          03032               btfsc          LCDTEMP1,2
064A   1696          03033               bsf            D2C
064B   19F0          03034               btfsc          LCDTEMP1,3
064C   149A          03035               bsf            D2D
064D   1A70          03036               btfsc          LCDTEMP1,4
064E   1716          03037               bsf            D2E
064F   1AF0          03038               btfsc          LCDTEMP1,5
0650   1712          03039               bsf            D2F
0651   1B70          03040               btfsc          LCDTEMP1,6
0652   1496          03041               bsf            D2G
0653   1303          03042               bcf            STATUS,RP1     ; Select bank 0
0654   0008          03043               return
                     03044
0655                 03045 LoadD3        ; Write to LCD digit 3
0655   00F0          03046               movwf          LCDTEMP1
```

```
0656   27B0           03047                call           Get7SegDat     ; get seven segment data in w
0657   00F0           03048                movwf          LCDTEMP1       ; save in temp
0658   1703           03049                bsf            STATUS,RP1     ; Select bank 2
0659   1311           03050                bcf            D3A
065A   1391           03051                bcf            D3B
065B   1395           03052                bcf            D3C
065C   1319           03053                bcf            D3D
065D   1016           03054                bcf            D3E
065E   1012           03055                bcf            D3F
065F   1315           03056                bcf            D3G
0660   1870           03057                btfsc          LCDTEMP1,0     ; if not set, skip segment
0661   1711           03058                bsf            D3A
0662   18F0           03059                btfsc          LCDTEMP1,1
0663   1791           03060                bsf            D3B
0664   1970           03061                btfsc          LCDTEMP1,2
0665   1795           03062                bsf            D3C
0666   19F0           03063                btfsc          LCDTEMP1,3
0667   1719           03064                bsf            D3D
0668   1A70           03065                btfsc          LCDTEMP1,4
0669   1416           03066                bsf            D3E
066A   1AF0           03067                btfsc          LCDTEMP1,5
066B   1412           03068                bsf            D3F
066C   1B70           03069                btfsc          LCDTEMP1,6
066D   1715           03070                bsf            D3G
066E   1303           03071                bcf            STATUS,RP1     ; Select bank 0
066F   0008           03072                return
                      03073
0670                  03074 LoadD4         ; Write to LCD digit 4
0670   00F0           03075                movwf          LCDTEMP1
0671   27B0           03076                call           Get7SegDat     ; get seven segment data in w
0672   00F0           03077                movwf          LCDTEMP1       ; save in temp
0673   1703           03078                bsf            STATUS,RP1     ; Select bank 2
0674   1211           03079                bcf            D4A
0675   1291           03080                bcf            D4B
0676   1295           03081                bcf            D4C
0677   1219           03082                bcf            D4D
0678   1396           03083                bcf            D4E
0679   1392           03084                bcf            D4F
067A   1215           03085                bcf            D4G
067B   1870           03086                btfsc          LCDTEMP1,0     ; if not set, skip segment
067C   1611           03087                bsf            D4A
067D   18F0           03088                btfsc          LCDTEMP1,1
067E   1691           03089                bsf            D4B
067F   1970           03090                btfsc          LCDTEMP1,2
0680   1695           03091                bsf            D4C
0681   19F0           03092                btfsc          LCDTEMP1,3
```

```
0682   1619            03093                bsf             D4D
0683   1A70            03094                btfsc           LCDTEMP1,4
0684   1796            03095                bsf             D4E
0685   1AF0            03096                btfsc           LCDTEMP1,5
0686   1792            03097                bsf             D4F
0687   1B70            03098                btfsc           LCDTEMP1,6
0688   1615            03099                bsf             D4G
0689   1303            03100                bcf             STATUS,RP1      ; Select bank 0
068A   0008            03101                return
                       03102
068B                   03103 LoadD5         ; Write to LCD digit 5
068B   00F0            03104                movwf           LCDTEMP1
068C   27B0            03105                call            Get7SegDat      ; get seven segment data in w
068D   00F0            03106                movwf           LCDTEMP1        ; save in temp
068E   1703            03107                bsf             STATUS,RP1      ; Select bank 2
068F   1091            03108                bcf             D5A
0690   1111            03109                bcf             D5B
0691   1115            03110                bcf             D5C
0692   1099            03111                bcf             D5D
0693   1195            03112                bcf             D5E
0694   1191            03113                bcf             D5F
0695   1095            03114                bcf             D5G
0696   1870            03115                btfsc           LCDTEMP1,0      ; if not set, skip segment
0697   1491            03116                bsf             D5A
0698   18F0            03117                btfsc           LCDTEMP1,1
0699   1511            03118                bsf             D5B
069A   1970            03119                btfsc           LCDTEMP1,2
069B   1515            03120                bsf             D5C
069C   19F0            03121                btfsc           LCDTEMP1,3
069D   1499            03122                bsf             D5D
069E   1A70            03123                btfsc           LCDTEMP1,4
069F   1595            03124                bsf             D5E
06A0   1AF0            03125                btfsc           LCDTEMP1,5
06A1   1591            03126                bsf             D5F
06A2   1B70            03127                btfsc           LCDTEMP1,6
06A3   1495            03128                bsf             D5G
06A4   1303            03129                bcf             STATUS,RP1      ; Select bank 0
06A5   0008            03130                return
                       03131
06A6                   03132 LoadD6         ; Write to LCD digit 6
06A6   00F0            03133                movwf           LCDTEMP1
06A7   27B0            03134                call            Get7SegDat      ; get seven segment data in w
06A8   00F0            03135                movwf           LCDTEMP1        ; save in temp
06A9   1703            03136                bsf             STATUS,RP1      ; Select bank 2
06AA   1310            03137                bcf             D6A
06AB   1390            03138                bcf             D6B
```

```
06AC   1394         03139              bcf          D6C
06AD   1318         03140              bcf          D6D
06AE   1015         03141              bcf          D6E
06AF   1011         03142              bcf          D6F
06B0   1314         03143              bcf          D6G
06B1   1870         03144              btfsc        LCDTEMP1,0      ; if not set, skip segment
06B2   1710         03145              bsf          D6A
06B3   18F0         03146              btfsc        LCDTEMP1,1
06B4   1790         03147              bsf          D6B
06B5   1970         03148              btfsc        LCDTEMP1,2
06B6   1794         03149              bsf          D6C
06B7   19F0         03150              btfsc        LCDTEMP1,3
06B8   1718         03151              bsf          D6D
06B9   1A70         03152              btfsc        LCDTEMP1,4
06BA   1415         03153              bsf          D6E
06BB   1AF0         03154              btfsc        LCDTEMP1,5
06BC   1411         03155              bsf          D6F
06BD   1B70         03156              btfsc        LCDTEMP1,6
06BE   1714         03157              bsf          D6G
06BF   1303         03158              bcf          STATUS,RP1      ; Select bank 0
06C0   0008         03159              return
                    03160
06C1                03161 LoadD7       ; Write to LCD digit 7
06C1   00F0         03162              movwf        LCDTEMP1
06C2   27B0         03163              call         Get7SegDat      ; get seven segment data in w
06C3   00F0         03164              movwf        LCDTEMP1        ; save in temp
06C4   1703         03165              bsf          STATUS,RP1      ; Select bank 2
06C5   1190         03166              bcf          D7A
06C6   1210         03167              bcf          D7B
06C7   1214         03168              bcf          D7C
06C8   1198         03169              bcf          D7D
06C9   1294         03170              bcf          D7E
06CA   1290         03171              bcf          D7F
06CB   1194         03172              bcf          D7G
06CC   1870         03173              btfsc        LCDTEMP1,0      ; if not set, skip segment
06CD   1590         03174              bsf          D7A
06CE   18F0         03175              btfsc        LCDTEMP1,1
06CF   1610         03176              bsf          D7B
06D0   1970         03177              btfsc        LCDTEMP1,2
06D1   1614         03178              bsf          D7C
06D2   19F0         03179              btfsc        LCDTEMP1,3
06D3   1598         03180              bsf          D7D
06D4   1A70         03181              btfsc        LCDTEMP1,4
06D5   1694         03182              bsf          D7E
06D6   1AF0         03183              btfsc        LCDTEMP1,5
06D7   1690         03184              bsf          D7F
```

```
06D8   1B70          03185                btfsc          LCDTEMP1,6
06D9   1594          03186                bsf            D7G
06DA   1303          03187                bcf            STATUS,RP1     ; Select bank 0
06DB   0008          03188                return
                     03189
06DC                 03190 LoadD8         ; Write to LCD digit 8 (right side)
06DC   00F0          03191                movwf          LCDTEMP1
06DD   27B0          03192                call           Get7SegDat     ; get seven segment data in w
06DE   00F0          03193                movwf          LCDTEMP1       ; save in temp
06DF   1703          03194                bsf            STATUS,RP1     ; Select bank 2
06E0   1010          03195                bcf            D8A
06E1   1090          03196                bcf            D8B
06E2   1094          03197                bcf            D8C
06E3   1018          03198                bcf            D8D
06E4   1114          03199                bcf            D8E
06E5   1110          03200                bcf            D8F
06E6   1014          03201                bcf            D8G
06E7   1870          03202                btfsc          LCDTEMP1,0     ; if not set, skip segment
06E8   1410          03203                bsf            D8A
06E9   18F0          03204                btfsc          LCDTEMP1,1
06EA   1490          03205                bsf            D8B
06EB   1970          03206                btfsc          LCDTEMP1,2
06EC   1494          03207                bsf            D8C
06ED   19F0          03208                btfsc          LCDTEMP1,3
06EE   1418          03209                bsf            D8D
06EF   1A70          03210                btfsc          LCDTEMP1,4
06F0   1514          03211                bsf            D8E
06F1   1AF0          03212                btfsc          LCDTEMP1,5
06F2   1510          03213                bsf            D8F
06F3   1B70          03214                btfsc          LCDTEMP1,6
06F4   1414          03215                bsf            D8G
06F5   1303          03216                bcf            STATUS,RP1     ; Select bank 0
06F6   0008          03217                return
                     03218
                     03219 ;*******************************************************************************
                     03220 ; Writes HEX values to LCD
                     03221 ;
                     03222 ; This routine is used to write the contents of the receive buffer to the LCD
                     03223 ; as a 6 digit HEX number
                     03224 ;
                     03225 ; This can also be used as a diagnostic tool. Copy the data to display to the
                     03226 ; receive buffer, and call this subroutine.
                     03227 ;*******************************************************************************
                     03228 ; Inputs: RXDATA0, RXDATA1, RXDATA2
                     03229 ; Ouputs: none
                     03230 ; Uses:   none
```

```
06F7                WriteLCD
06F7   2612                      call       ClrLCD        ; blank display

06F8   0839                      movf       RXDATA0,W     ; get data from receive buffer
06F9   390F                      andlw      0x0f
06FA   263A                      call       LoadD2
06FB   0E39                      swapf      RXDATA0,W
06FC   390F                      andlw      0x0f
06FD   261F                      call       LoadD1

06FE   083A                      movf       RXDATA1,W
06FF   390F                      andlw      0x0f
0700   268B                      call       LoadD5
0701   0E3A                      swapf      RXDATA1,W
0702   390F                      andlw      0x0f
0703   2670                      call       LoadD4

0704   083B                      movf       RXDATA2,W
0705   390F                      andlw      0x0f
0706   26DC                      call       LoadD8
0707   0E3B                      swapf      RXDATA2,W
0708   390F                      andlw      0x0f
0709   26C1                      call       LoadD7

070A   0008                      return

                    ;***************************************************************************
                    ; Writes Decimal values to LCD
                    ; Takes packed BCD numbers in R0, R1, R2 and writes each digit to LCD
                    ; Write R0 to digit 4
                    ; Write R1 to digits 5 and 6
                    ; Write R2 to digits 7 and 8
                    ;***************************************************************************
                    ; Inputs: R0, R1, R2
                    ; Ouputs: none


070B                WriteLCD2
070B   2612                      call       ClrLCD        ; clear the LCD display

070C   084E         WriteLCD4    movf       R0,W          ; get R0
070D   390F                      andlw      0x0f          ; mask off upper nibble
070E   2670                      call       LoadD4        ; LCD digit 4

070F   0E4F         WriteLCD5    swapf      R1,W          ; swap R1 into W
```

```
0710   390F          03277                andlw          0x0f           ; mask off upper nibble (former lower nibble)
0711   268B          03278                call           LoadD5         ; LCD digit 5
                     03279
0712   084F          03280 WriteLCD6      movf           R1,W           ; get BCD in R1
0713   390F          03281                andlw          0x0f           ; mask off upper nibble
0714   26A6          03282                call           LoadD6         ; LCD digit 6
                     03283
0715   0E50          03284 WriteLCD7      swapf          R2,W           ; swap R2 into W
0716   390F          03285                andlw          0x0f           ; mask off upper nibble
0717   26C1          03286                call           LoadD7         ; LCD digit 7
                     03287
0718   0850          03288 WriteLCD8      movf           R2,W           ; get BCD in R2
0719   390F          03289                andlw          0x0f           ; mask off upper nibble
071A   26DC          03290                call           LoadD8         ; LCD digit 8
                     03291
071B   0008          03292                return
                     03293
                     03294 ;*****************************************************************************
                     03295 ; Displays packed BCD digits R0, R1, and R2, leading zeros are not displayed.
                     03296 ; Called by APLoop, TPLoop and WHLoop
                     03297 ;*****************************************************************************
                     03298 ; Inputs: R0, R1, R2
                     03299 ; Ouputs: none
                     03300
                     03301
071C   084E          03302 DispPwr        movf           R0,W                        ; if "10K" digit is 0, leave blank
071D   390F          03303                andlw          0x0F
071E   1903          03304                btfsc          STATUS,Z
071F   2F22          03305                goto           DP5LCD
0720   270C          03306                call           WriteLCD4                   ; Display 5 digits
0721   0008          03307                return
                     03308
0722   0E4F          03309 DP5LCD         swapf          R1,W                        ; if "1K" digit is zero, leave blank
0723   390F          03310                andlw          0x0F
0724   1903          03311                btfsc          STATUS,Z
0725   2F28          03312                goto           DP4LCD
0726   270F          03313                call           WriteLCD5                   ; Display 4 digits
0727   0008          03314                return
                     03315
0728   084F          03316 DP4LCD         movf           R1,W                        ; if "100" digit is zero, leave blank
0729   390F          03317                andlw          0x0F
072A   1903          03318                btfsc          STATUS,Z
072B   2F2E          03319                goto           DP3LCD
072C   2712          03320                call           WriteLCD6                   ; Display 3 digits
072D   0008          03321                return
                     03322
```

```
072E   0E50          03323 DP3LCD         swapf          R2,W                        ; if "10" digit is zero, leave blank
072F   390F          03324                andlw          0x0F
0730   1903          03325                btfsc          STATUS,Z
0731   2F34          03326                goto           DP2LCD
0732   2715          03327                call           WriteLCD7                   ; Display 2 digits
0733   0008          03328                return
                     03329
0734   2718          03330 DP2LCD         call           WriteLCD8                   ; Display 1 digit
0735   0008          03331                return
                     03332
07B0                 03333                org 0x07B0
                     03334 ;*****************************************************************************
                     03335 ; LCD character table
                     03336 ;
                     03337 ; Place the character to be displayed on the LCD in LCDTEMP1, and call this routine.
                     03338 ; The seven segment pattern is returned in W.
                     03339 ;
                     03340 ; Place LCD character table so that a memory 256 byte boundry does not occur
                     03341 ; within the table itself.
                     03342 ;
                     03343 ; Entering this routine with LCDTEMP1=0x1B returns the device serial number rather
                     03344 ; than a seven segment display pattern.
                     03345 ;*****************************************************************************
                     03346 ; Inputs: LCDTEMP1, PCL
                     03347 ; Ouputs: W
                     03348 ; Uses:   PCLATH
                     03349
07B0                 03350 Get7SegDat
07B0   3007          03351                movlw          high(Get7SegDat)
07B1   008A          03352                movwf          PCLATH         ; set PCLATH to jump to this table
                     03353
07B2   0870          03354                movf           LCDTEMP1,W     ; get offset
07B3   3C1D          03355                sublw          0x1D           ; compare with table length
07B4   1C03          03356                btfss          STATUS,C       ; was there a borrow?
07B5   3440          03357                retlw          0x40           ; yes, return dash
                     03358
07B6   0870          03359                movf           LCDTEMP1,W     ; get offset
07B7   0782          03360                addwf          PCL,F          ; add to program counter
07B8   343F          03361                retlw          b'00111111'    ; Zero
07B9   3406          03362                retlw          b'00000110'    ; One
07BA   345B          03363                retlw          b'01011011'    ; Two
07BB   344F          03364                retlw          b'01001111'    ; Three
07BC   3466          03365                retlw          b'01100110'    ; Four
07BD   346D          03366                retlw          b'01101101'    ; Five
07BE   347D          03367                retlw          b'01111101'    ; Six
07BF   3407          03368                retlw          b'00000111'    ; Seven
```

```
07C0   347F           03369                retlw          b'01111111'    ; Eight
07C1   346F           03370                retlw          b'01101111'    ; Nine
07C2   3477           03371                retlw          b'01110111'    ; A
07C3   347C           03372                retlw          b'01111100'    ; b
07C4   3439           03373                retlw          b'00111001'    ; C
07C5   345E           03374                retlw          b'01011110'    ; d
07C6   3479           03375                retlw          b'01111001'    ; E
07C7   3471           03376                retlw          b'01110001'    ; F
07C8   3473           03377                retlw          b'01110011'    ; P (0x10)
07C9   343C           03378                retlw          b'00111100'    ; left side of W (0x11) "Wl"
07CA   341E           03379                retlw          b'00011110'    ; right side of W (0x12) "Wr"
07CB   3478           03380                retlw          b'01111000'    ; t (0x13)
07CC   3450           03381                retlw          b'01010000'    ; r (0x14)
07CD   345C           03382                retlw          b'01011100'    ; o (0x15)
07CE   341C           03383                retlw          b'00011100'    ; u (0x16)
07CF   3400           03384                retlw          b'00000000'    ; space (0x17)
07D0   3476           03385                retlw          b'01110110'    ; H (0x18)
07D1   3433           03386                retlw          b'00110011'    ; left side of M (0x19) "Ml"
07D2   3427           03387                retlw          b'00100111'    ; right side of M (0x1A) "Mr"
07D3   0835           03388                movf           SERNUM,W       ; serial number (0x1B)
07D4   0008           03389                return                        ; (0x1C)
07D5   3438           03390                retlw          b'00111000'    ; L (0x1D)
                      03391
                      03392 ;*****************************************************************************
                      03393 ; Characters used in the initial scrolling message listed in reverse order.
                      03394 ; The DT directive generates a group of RETLW statements with the contents of
                      03395 ; the table. This is a more compact way of specifying the table contents than
                      03396 ; listing "RETLW 0xNN" statements.
                      03397 ;*****************************************************************************
                      03398 ; Inputs: PTRTMP, PCL
                      03399 ; Ouputs: W
                      03400 ; Uses:   PCLATH
                      03401
07D6                  03402 Msg
07D6   3007           03403                movlw          high(Msg)
07D7   008A           03404                movwf          PCLATH         ; set PCLATH to jump to this table
                      03405
07D8   0834           03406                movf           PTRTMP,W       ; get offset
07D9   0782           03407                addwf          PCL,F
07DA   3400 3417 3417 03408                dt             0x00,0x17,0x17,0x1B,0x17,0x14,0x0E,0x13,0x0E,0x1A,0x19,0x17
       341B 3417 3414 03409 ;                               0   sp   sp   sn   sp   r    E    t    E    Mr   Ml   sp
       340E 3413 340E 03410
       341A 3419 3417
07E6   3414 3416 3415 03411                dt             0x14,0x16,0x15,0x18,0x17,0x13,0x13,0x0A,0x12,0x11,0x17,0x03
       3418 3417 3413 03412 ;                               r    u    o    H   sp   t    t    A    Wr   Wl  sp   3
       3413 340A 3412 03413
```

```
       3411 3417 3403
07F2   3402 3409 340C 03414                dt              0x02,0x09,0x0C,0x06,0x01,0x0C,0x01,0x10
       3406 3401 340C 03415 ;                               2    9    C    6    1    C    1    P
       3401 3410      03416
07FA   0000           03417                nop             ; marker to find end of tables
                      03418                                ; check if overrun to next page,
                      03419                                ; or tables cross 256 word boundry
                      03420
                      03421 ;*************************************************************************
                      03422 ; End of program code.
                      03423 ;
                      03424 ; The fill directive fills the second program memory page with "goto 0x00" instructions.
                      03425 ; At 0x000 in the second page is an instruction to clear PCLATH. The next goto instruction will
                      03426 ; then goto the first instruction in the first page.
                      03427 ;
                      03428 ; If execution should somehow branch to page 1, the next 3
                      03429 ; lines will force execution to the reset vector on page 0.
                      03430 ;
                      03431 ; This is not the same as a reset. Among other things, the stack is not reset.
                      03432
0800                  03433                org             0x0800          ; Page 1
0800   018A           03434                clrf            PCLATH          ; selects page 0
0801   28             03435                fill (goto 0x00),0x1000-$ ; jump to first instruction in whichever page is selected
                      03436
                      03437                end             ; directive indicating the end of code


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X---XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0300 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0340 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0380 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
03C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0400 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

```
0440 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0480 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
04C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0500 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0540 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0580 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
05C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0600 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0640 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0680 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
06C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0700 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXX----------
0780 : ---------------- ---------------- ---------------- XXXXXXXXXXXXXXXX
07C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX-----
0800 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0840 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0880 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
08C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0900 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0940 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0980 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
09C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0A00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0A40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0A80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0AC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0B00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0B40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0B80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0BC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0C00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0C40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0C80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0CC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0D00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0D40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0D80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0DC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0E00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0E40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0E80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0EC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0FC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

```
2000 : -------X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:  3966
Program Memory Words Free:   130


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,  2204 suppressed
```

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/01/02