NASA-CR-192722

SBIR-09.12-8442

Release Date: 3/13/94

p. 529

FINAL REPORT

A LOW-COST, CCD SOLID STATE STAR TRACKER

CONTRACT NAS5-31169

Applied Research Corporation
8201 Corporate Drive
Suite 1120
Landover, MD 20785

(NASA-CR-192722) A LOW-COST, CCD SOLID STATE STAR TRACKER Final Report, 1 Dec. 1990 - 30 Nov. 1992 (Applied Research Corp.) 529 p

N94-32417

Unclas

00/74 0010556

TABLE OF CONTENTS

OVERVIEW

The object of this Phase II SBIR project was to design, develop, fabricate, and test a flight-capable model of a low-cost CCD star tracker with multistar tracking capability. High update rate, low power, low mass, and sensitivity comparable to current NASA star trackers were desired in a design based on high reliability, radiation hard components. Mechanical and electrical interfaces must allow direct replacement of current NASA star tracker.

The star tracker incorporates a Texas Instrument TC217 CCD sensor, high-reliability and radiation hard electronics, and optical components fabricated out of radiation hard glasses. The engineering unit compares favorably in functional performance tests to the standard NASA star tracker. The CCD star tracker is characterized by a flexible architecture, allowing multiple operation modes (e.g., energy saving, high update, mapping, etc.) which are software selectable. To allow even greater flexibility, the star tracker software can be updated through a remote link. Thus, the tradeoffs between star tracker capabilities (update rate, resolution etc.) and power requirements can be fine-tuned for each mission.

The star tracker software was developed to the level of checking hardware functionality, demonstrating concept feasibility, and supporting performance tests. Development of hardware-interactive modules has been completed, while high level routines are in preliminary versions. The software is upgradable by remote link (RS-422). A hardware-protected kernel ensures communication link continuity during the upgrade process. Currently the star tracker performance is limited by single-tasking software. The hardware was designed to be able to perform several tasks simultaneously (exposure, data conversion, data analysis, reporting) using an interrupt-driven method of operation. Utilizing these capabilities with multitasking software will achieve a tracking rate limited only by sensor sensitivity.

In summary, ARC has designed, developed, and functionally tested an engineering model of a CCD star tracker with multistar tracking capabilities. Test results prove that the performance of the engineering model star tracker (even with suboptimal software) exceeds that of the standard NASA star tracker, which can track only a single star. Software upgrades to permit multi-tasking operation will further improve the performance of the CCD star tracker.

DESIGN CONCEPT

Design objectives were:

- multi-star tracking capability,
- low production cost,
- reliability,
- low power consumption, and
- mechanical and electrical compatibility with standard NASA star tracker.

Above objectives, together with limited development time and budget, imposed design constraints and forced certain trade-offs.

The most severe impact on design selection was low production cost requirements. It required:

- transmission optics in place of mirrors, and
- commercial CCD sensor in place of astronomical grade CCDs.

The last selection was additionally supported by the low power requirement, (thus necessity of selecting a sensor capable of operating in the temperature range -10°C to +50°C without cooling, or with only minimal cooling in high temperature environments.

Selecting a sensor capable of rapid (0.5ms) image transfer into internal sensor memory avoids a mechanical shutter, thus decreasing cost and power requirements, and increasing reliability by avoiding movable mechanical components.

Reliability requirement has been fulfilled by selecting:

- electronics components from PPL19, ISTP and Sampex - high reliability components list.

In addition all parts have been derated (at least by 20%) with respect to producer specification. However, restricted component selection results in limited CPU processing power, and higher electronics mass and volume than with state of the art components. Some alternate designs are listed at the section titled "Suggested Improvements".

Low power consumption requirement have been fulfilled by:

- using CMOS digital chips,
- avoiding line termination,
- building all power drivers using complimentary MOSFETS,
- switching power off to all bipolar component when they are not operating, and
- implementing a "sleep" mode whenever possible.

As low power requirements cannot be fullfiled together with a high update rate, the star tracker was designed with a flexible hardware architecture and several operational modes. These modes can be selected by software, allowing tuning of the power requirement, update rate, and processing complexity for each particular mission.

Requirement of mechanical and electrical compatability with standard NASA star tracker forces external shape of star tracker and required placing the electronics components on PCB's of a peculiar shape. As electrical compatibility may not always be an issue, the star tracker interface to external connector was implemented as a separate module - allowing easy modification.

Remaining electronics were also implemented in modular fashion allowing easy upgrades (adding memory board, replacing CPU, etc.).

Limited development time and budget determined software development and processor choice. A PC was chosen as the cross development platform due to programming tools availability, low cost and ARC staff experience.

DESIGN SUMMARY

CCD star tracker features:

Lens: 50mm, f/1.4, field of view of 8°x 8° Six element Petzval lens compensated for 600-1000nm. Built of rad hard glass. Athermalized in -10°C to +50°C range. Optical losses 15% (T/1.54).

Sensor: CCD Texas Instruments TC217 1134 x 486 (interlaced 972). Pixels 7.8 x 13.6 μm^2. Image area 8.84 x 6.6 mm^2 (11.7mm diag.). Multiple analog storage area, electronic shutter (about 0.5ms) QDE 40% in the spectral range 500-850nm. NES 15 electrons. Dynamic range 60dB min. Field of view 10.1° x 7.55°; area 76.26° square. Pixel field of view 32" x 56" (26" interlaced).

Electronics: Radiation hard components, conservative design. CPU 80C82, 128K RAM, 128K EEPROM Flexible architecture - software controllable parallel operation synchronized by custom sequencer, CPU and timers. CPU clock 4MHz (20% derated).

Mechanical: Mechanical compatibility with NASA standard star tracker (Ball Brothers).

Interface: RS-422 full duplex (4 wires). Power on baud rate 9600. Command switchable baud rates 19.2K, 38.4K, *80K. Analog x, y, m, (+/- 5V 500Ω) star presence (0 or +5V 220Ω). Two external control signals (0, +28V or 0, +5V with jumper removed), can be software reconfigurated (e.g. acquisition start, reduced search). Compatible with NASA standard star tracker.
*For testing only - excess UART specification.

PERFORMANCE SUMMARY

CCD star tracker performance:

Field of view 10° x 7.5°

Spectral range 500-1000nm, 600-800nm max sensitivity

Noise equivalent angle for brightest star in field of view
3 x 3 arc second

Point spread (FWHH)
4 x 4 pixels (interlaced)
Full width 10% above noise level
8 x 8 pixels

Optical system aberations

	for 4° x 4° FOV	for 8° x 8° FOV
Monochromatic (850nm)	< 1 pixel (30")	< 2 pixels (60")
1000 - 600nm extreme	< 2 pixels (60")	< 4 pixels (120")

Subpixel interpolation error due to optics astigmatism and coma aberrations

4" x 4° FOV	8° x 8° FOV
< 1 pixel (30")	< 2 pixels (60")

Subpixel interpolation linearity
5 x 5 arc second (without software processing)

Power requirements
1-3W mode dependent

Sensitivity vs update rate

Update Rate [H_2]	Sensitivity [Star Magnitude]
1	5-6
*5	4
*20	2-3

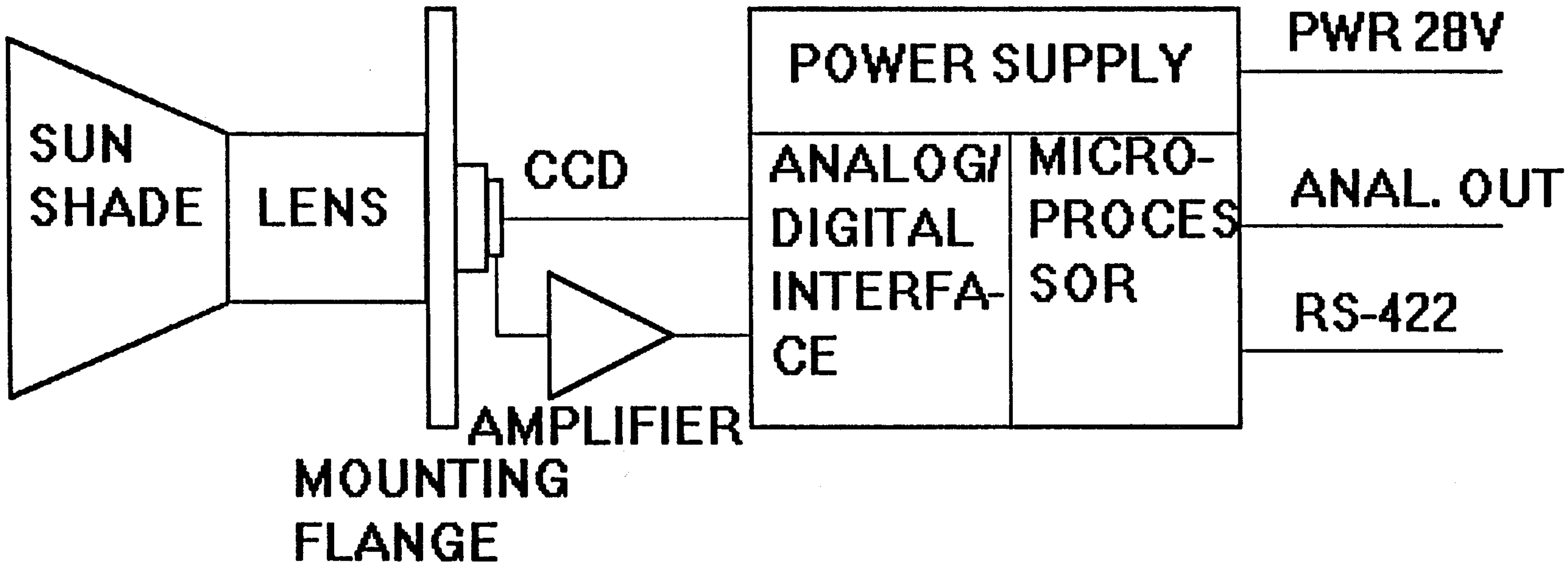
* Required software improvement

Acquisition time
* 1 sec

* Software module not fully tested

Power on ready
0.5 sec

STAR TRACKER BLOCK DIAGRAM



PHYSICAL DESIGN CHARACTERISTICS

OPTICS

Custom designed by Tuscon Optical Research Corporation.

50.3mm at f/1.4

8° x 8° field of view at 600-1000mm

Petzval 6 element lens built of rad hard optical glass (Schott SF6G05 and SSK5G06)

Single magnesium fluoride coating

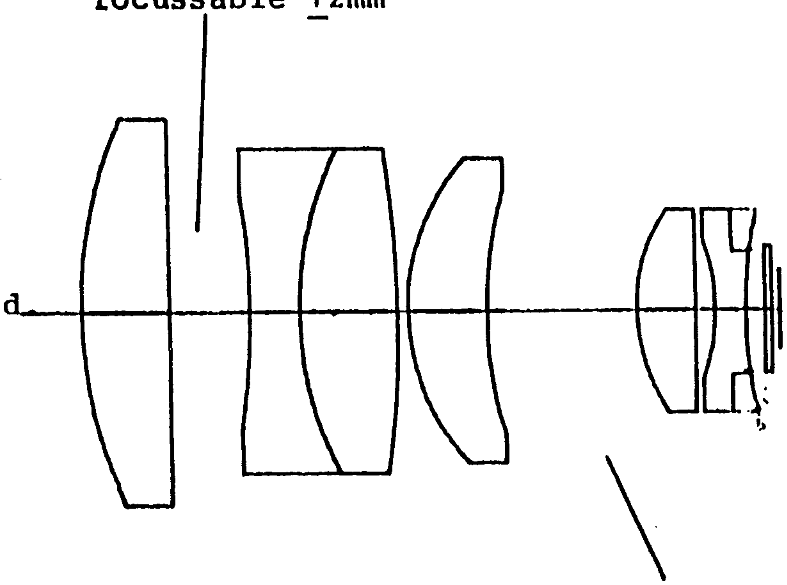
Optical losses 15%

Athermalized in -10°C to +50° range

Scale factor change .0004 in -10°C to +50°C corresponds to 1µm at the image edge

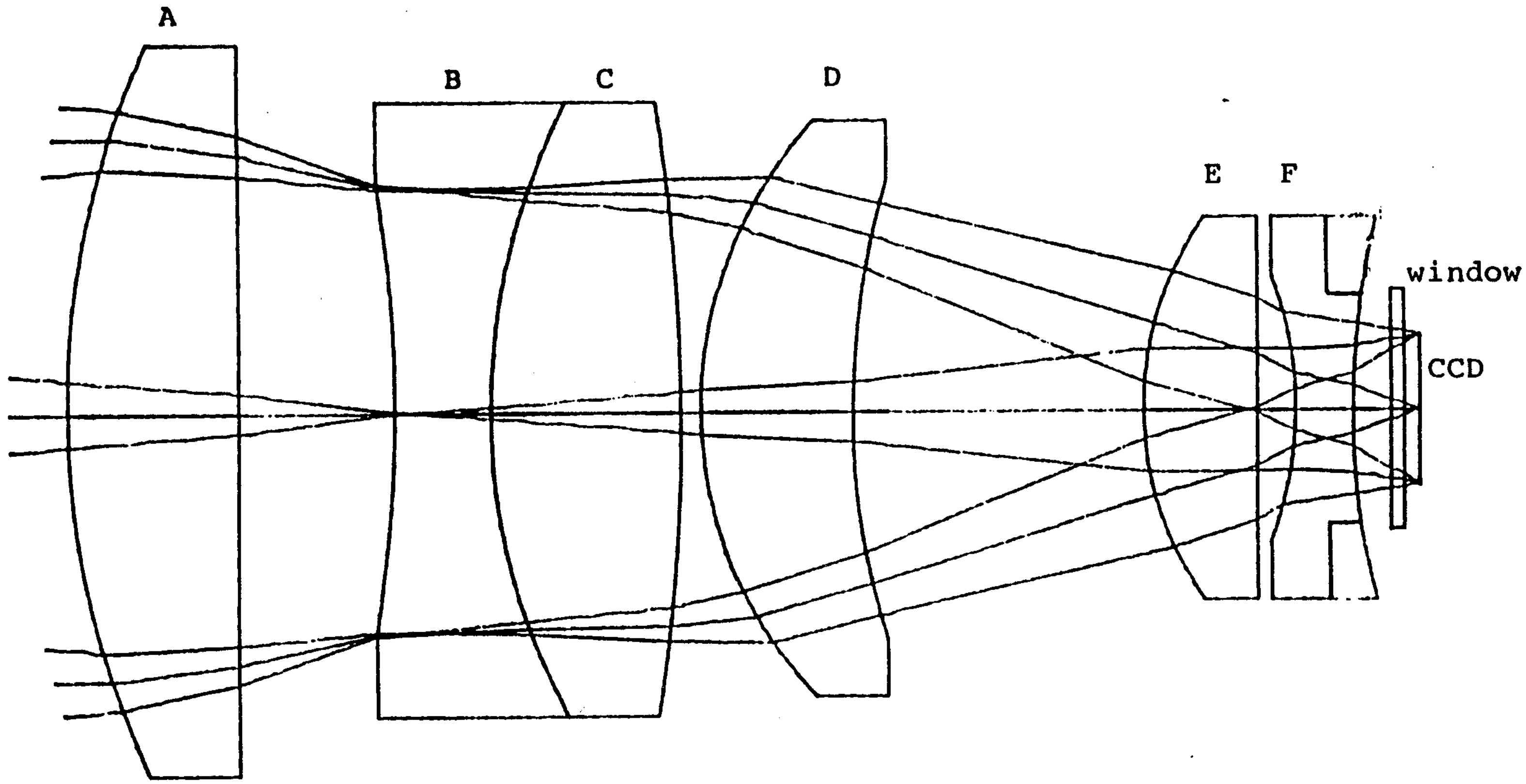
Temperature range: -10 to +50°C

Lens must be initially focussed at 20-29°C for infinity target.



Cell made from AL 6061 or similar coeff. of expansion 23.4×10^{-6} /°C

STAR 50C ALL ALUM EXCEPT SP9=INVAR	REF. LENGTH 50.80



Scale 2X

Fig. 2. Unvignetted ray paths

50MM F/1.4 STAR TRACKER .6 -1.0 MICRON	REF. LENGTH 25.40

STAR SPOT SIZE

Intensity drops to 50% peak at distance of 1 pixel (7.8μm) from center, to 2% at distance of 2 pixels.

60% of total flux in 4 pixels

15% of total flux in 1 pixel (defocussing factor)

Calculated Flux Distribution

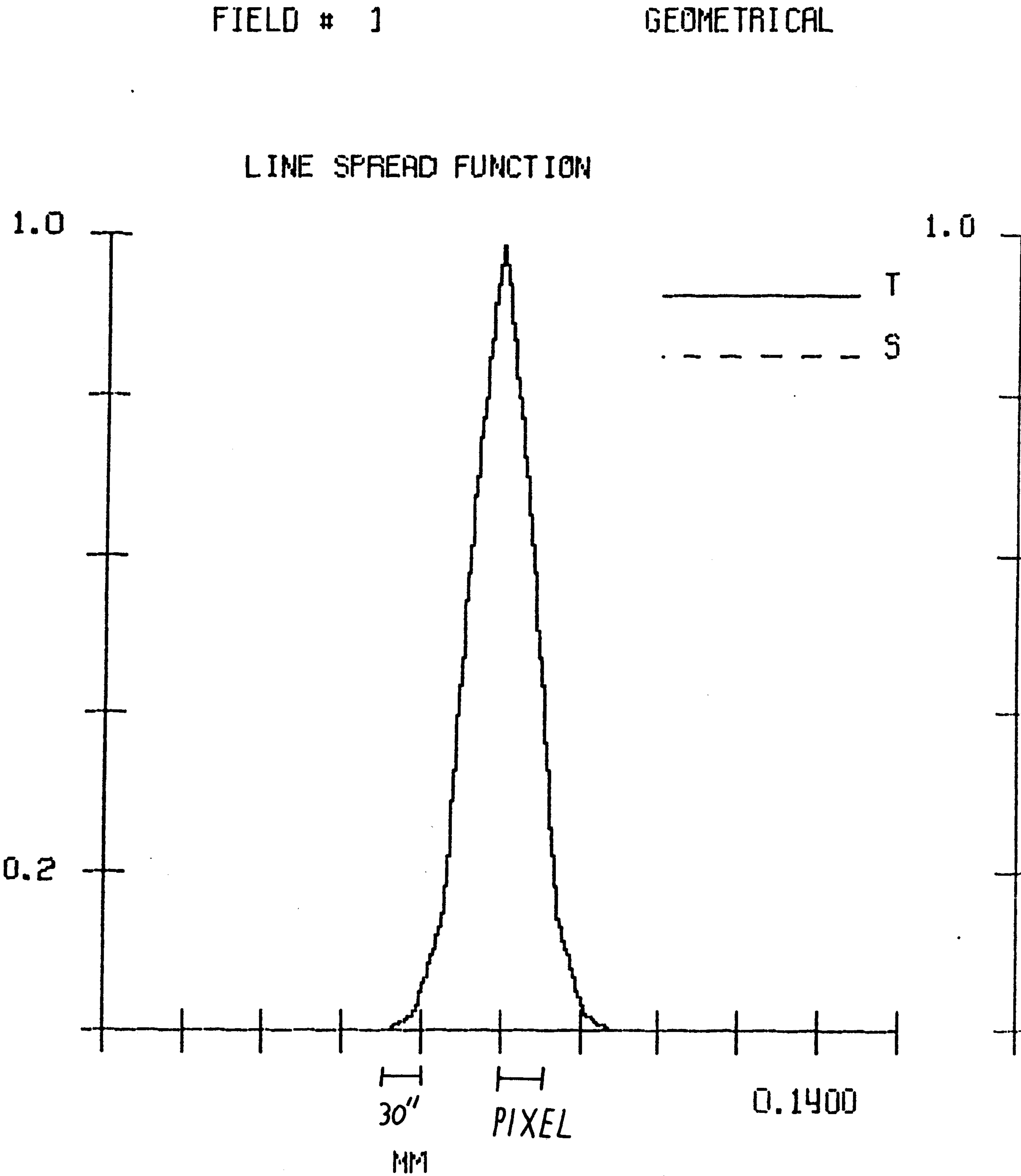
	5%	5%	
5%	15%	15%	5%
5%	15%	15%	5%
	5%	5%	

40% QDE and 15% Absorption => 5% of total flux in one pixel

Calculated Flux Intensity

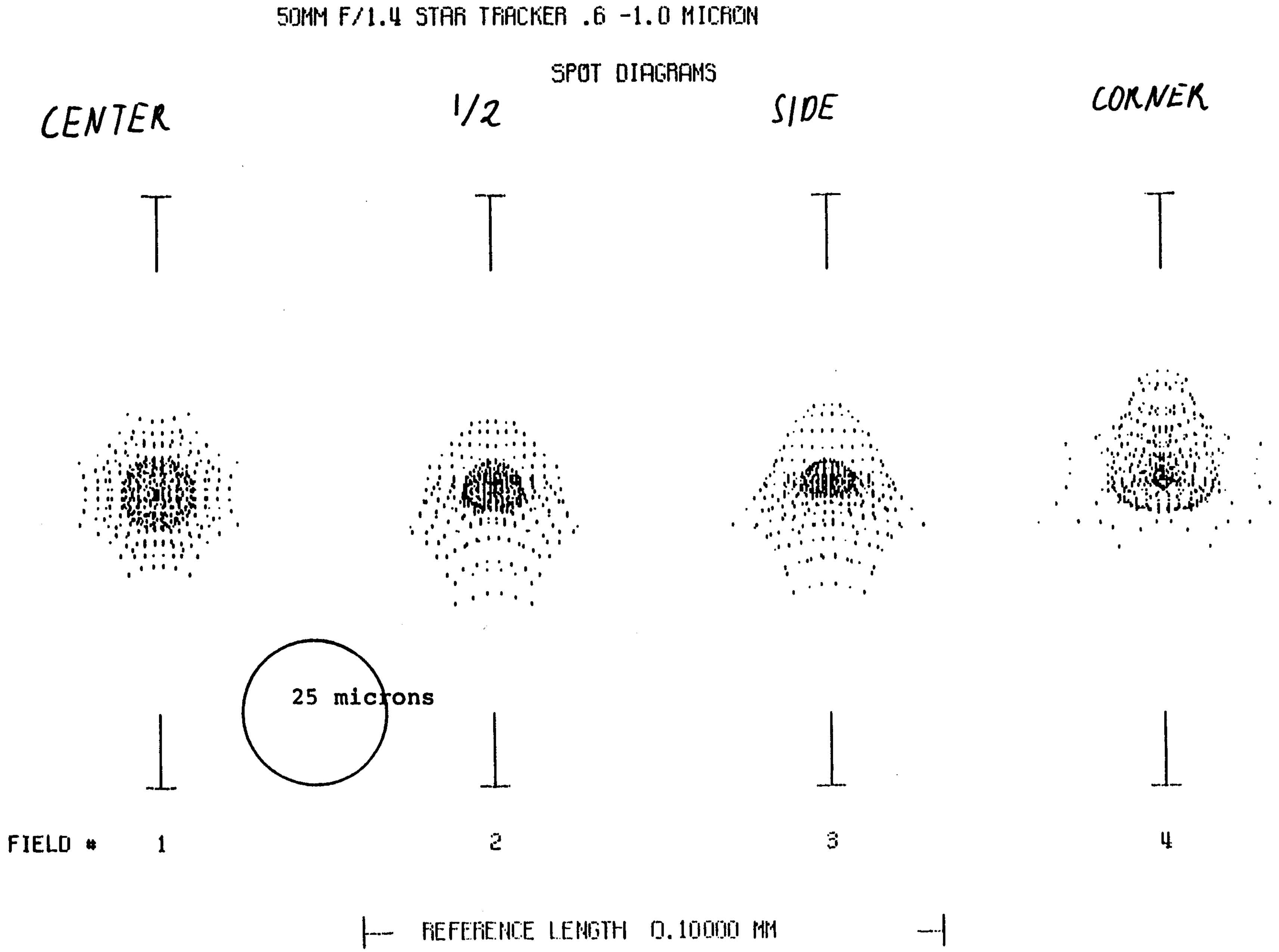
	2%	2%	
2%	5%	5%	2%
2%	5%	5%	2%
	2%	2%	

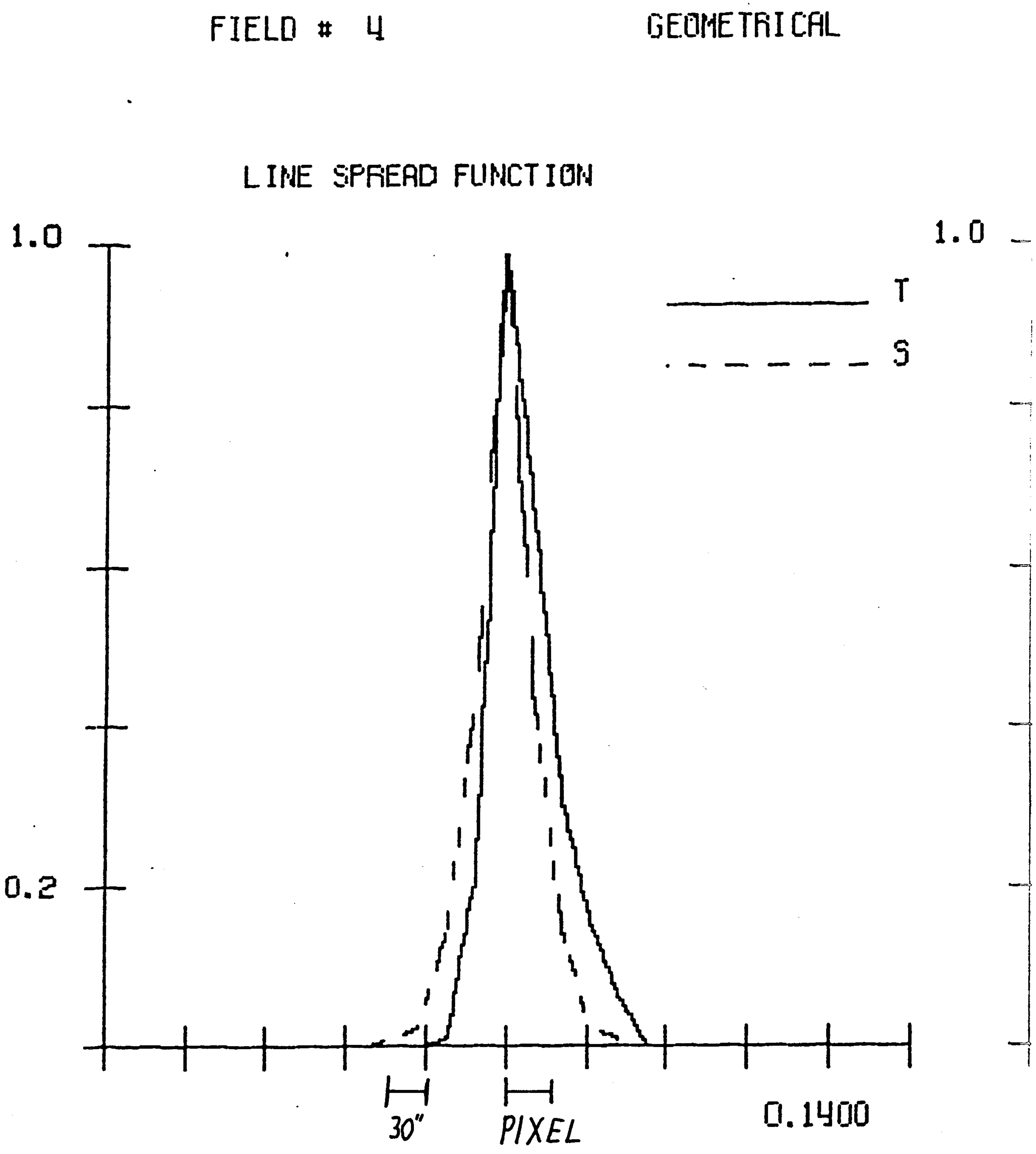
There is about 1/2 pixel asymmetry at the edge of the field of view which can be corrected by software improvements.



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

50MM F/1.4 STAR TRACKER .6 -1.0 MICRO


50MM F/1.4 STAR TRACKER .6 -1.0 MICRON
SPOT DIAGRAMS
CENTER
1/2
SIDE
CORNER
25 microns
FIELD # 1
2
3
4
REFERENCE LENGTH 0.10000 MM



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

50MM F/1.4 STAR TRACKER .6 -1.0 MICRO

CCD SENSOR

Considered --
- Tektronix: TK512
- Thomson-CSF: TH7883,TH7884,TH7863
- Texas Instruments: TI213, TI215, TI217
- Kodak: KAF-300L

Selected --
- Texas Instruments: TC217

Features of TC217 --
- 1134 x 486 (972 interlaced) pixels
- Each 7.8 x 13.6μm^2

Field of View with 50mm optic	Mode: 1134 x 486	Mode: 1134 x 972 interlaced
Entire CCD	10° x 7.5°	10° x 7.5°
1 pixel	32"(H) x 56"(V)	32"(H) x 28"(V)
Expected resolution at 1/10 of pixel	3" (H) x 6"(V)	3" x 3"

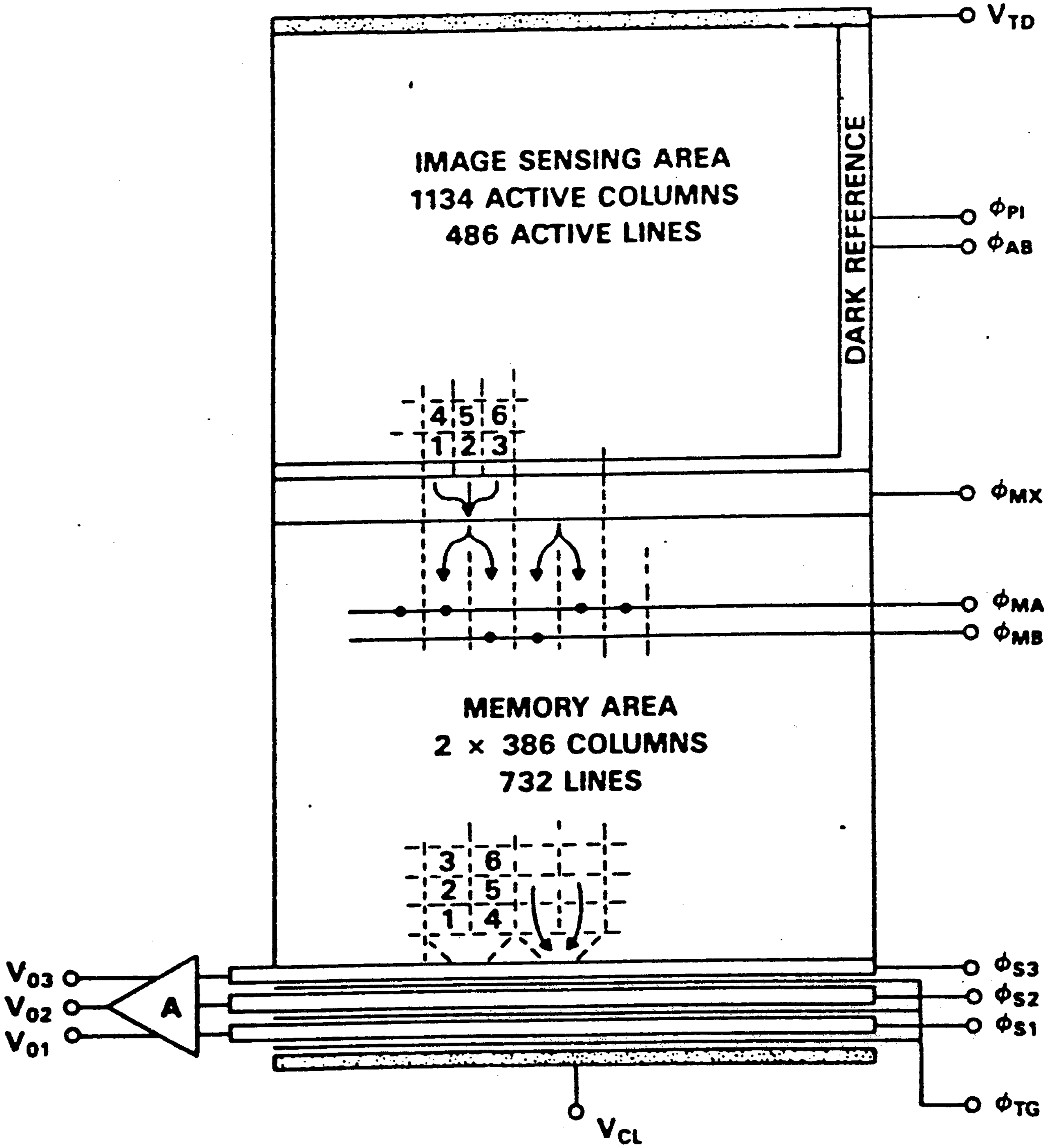
Dynamic range
- Full well 66,000 e^-
- Uniformity <1%
- Dark Current at RT 200electrons/sec/pixel

Conversion factor 6.2μV/electrons
QDE 40% in range 500-850nm

Architecture --
- Image area
- Two storage areas
- Shift registers
- Clearing drain

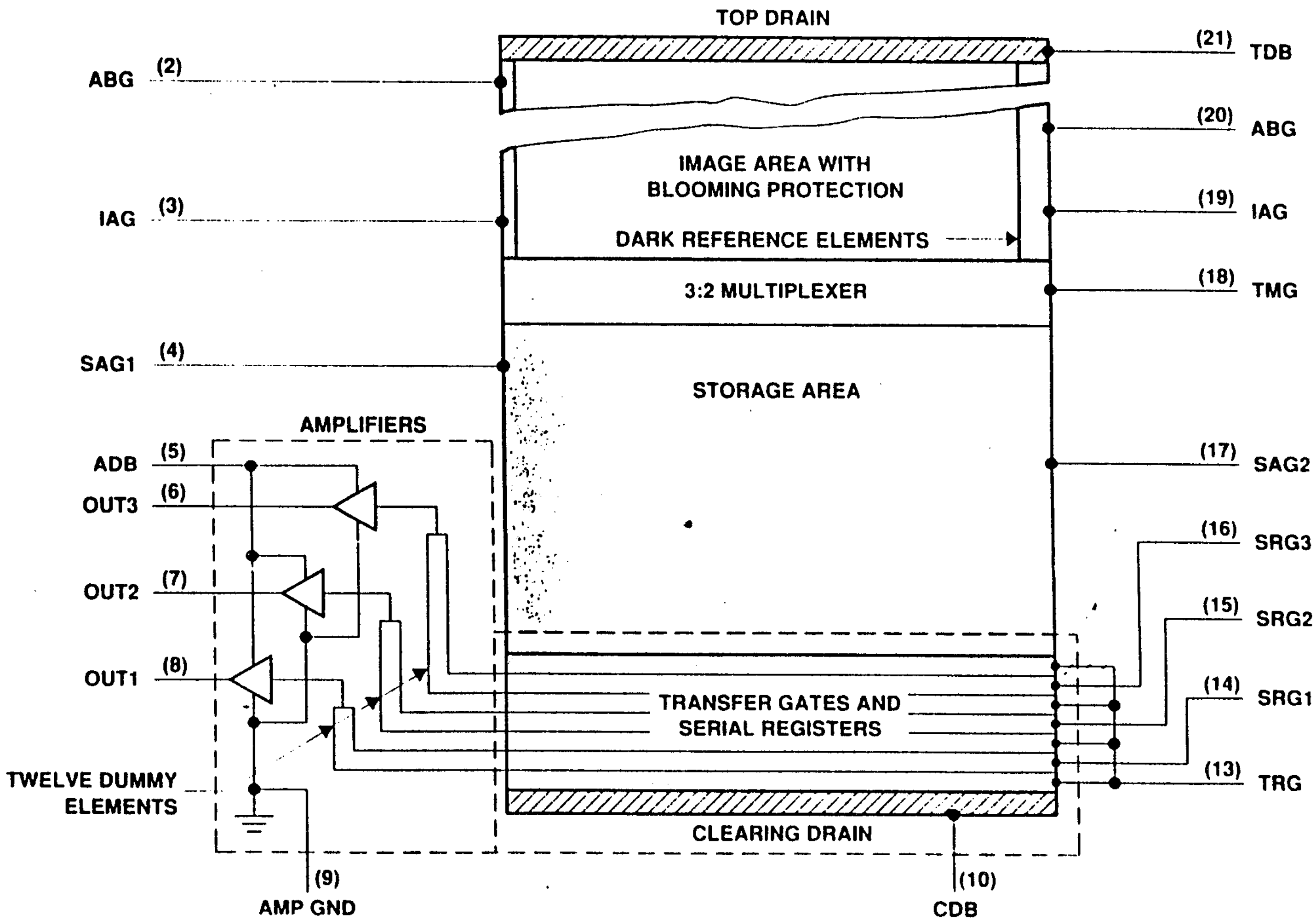
Advantages --
- Room temperature operation (very low dark current)
- Electronic shutter mode - image transferred to storage area in 0.7ms
- Exposure and data acquisition in parallel
- Clearing drain - fast dumping of irrelevant portion of image, i.e. fast access to selected image window
- Virtual phase shift registers - simplified clock signal

Disadvantages --
- Complicated architecture
- Large number of clock signals



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

functional block diagram



MECHANICAL DESIGN

Mechanical compatibility with NASA standard star tracker (Ball Brothers).

Thermal compensation:

- Athermalized lenses
- CCD attachment

CCD ATTACHMENT

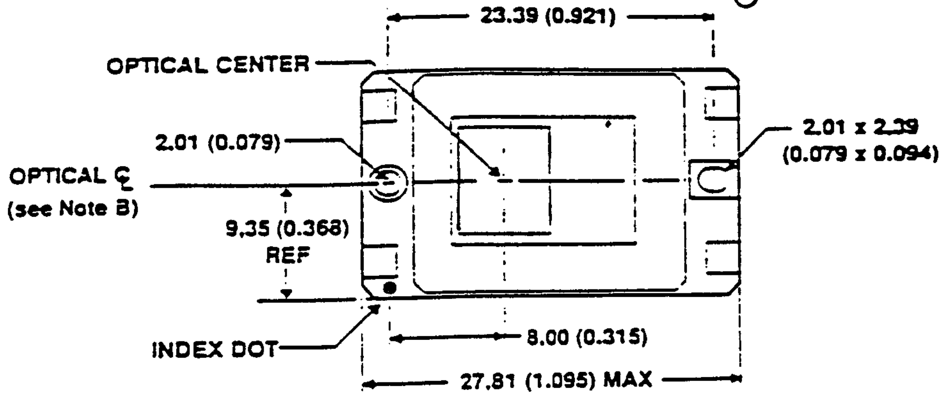
Asymmetrical position of image area with respect to CCD package requires thermal matching.

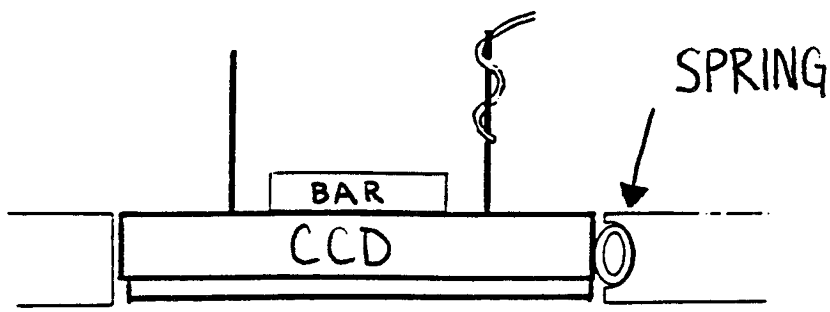
Thermal Expansion Coefficients

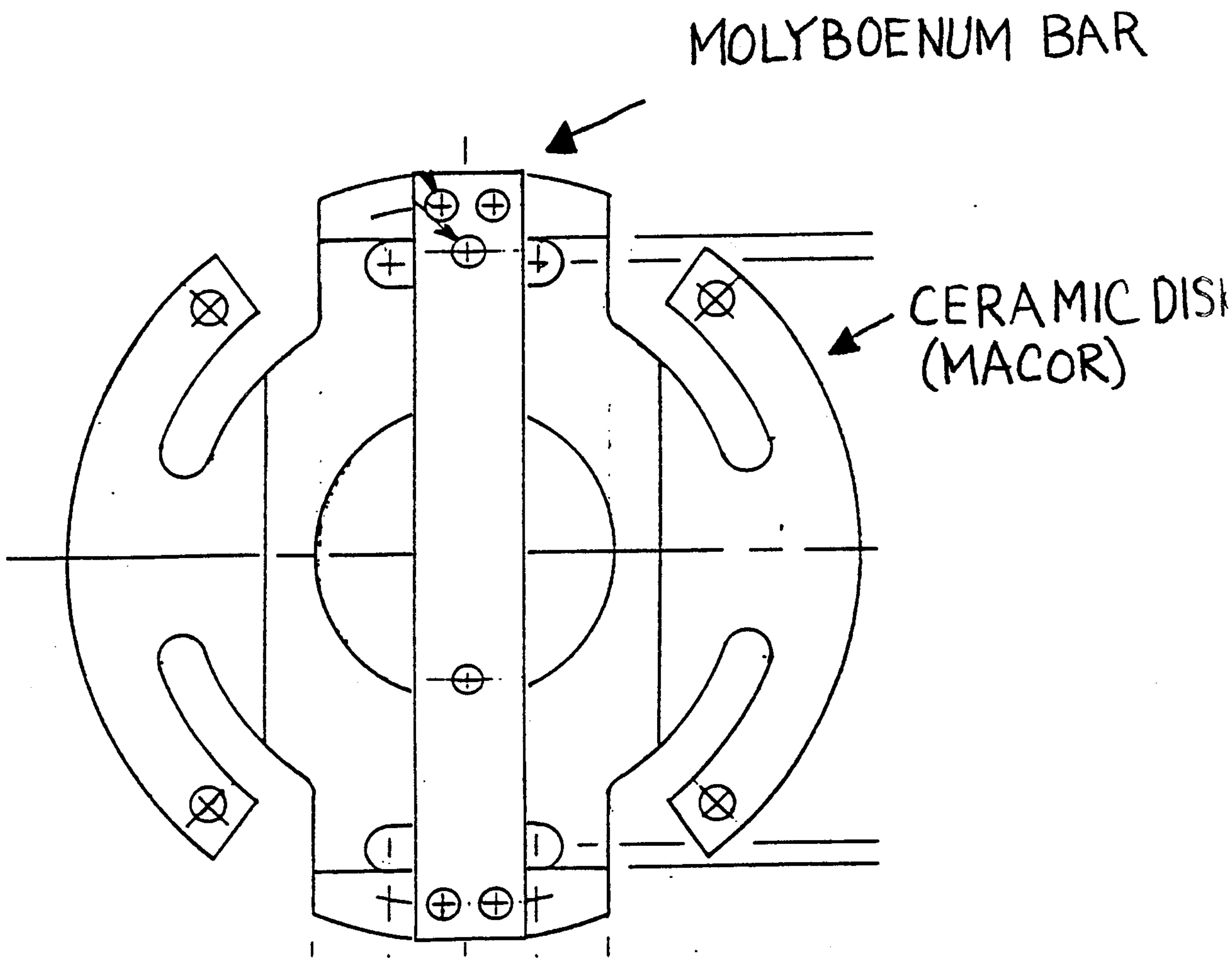
Material	Thermal Expansion Coefficient ppm/°C
Silicon	5±2.5
Kovar	5±1
Molybdenum	5±1
Ceramic (macor)	9
Aluminum	23±2

Molybdenum bar, ceramic disk and aluminum base plate are joined together systematically with respect to image center. This image center is not moving with temperature changes in spite of thermal coefficients mismatch.

CCD Package

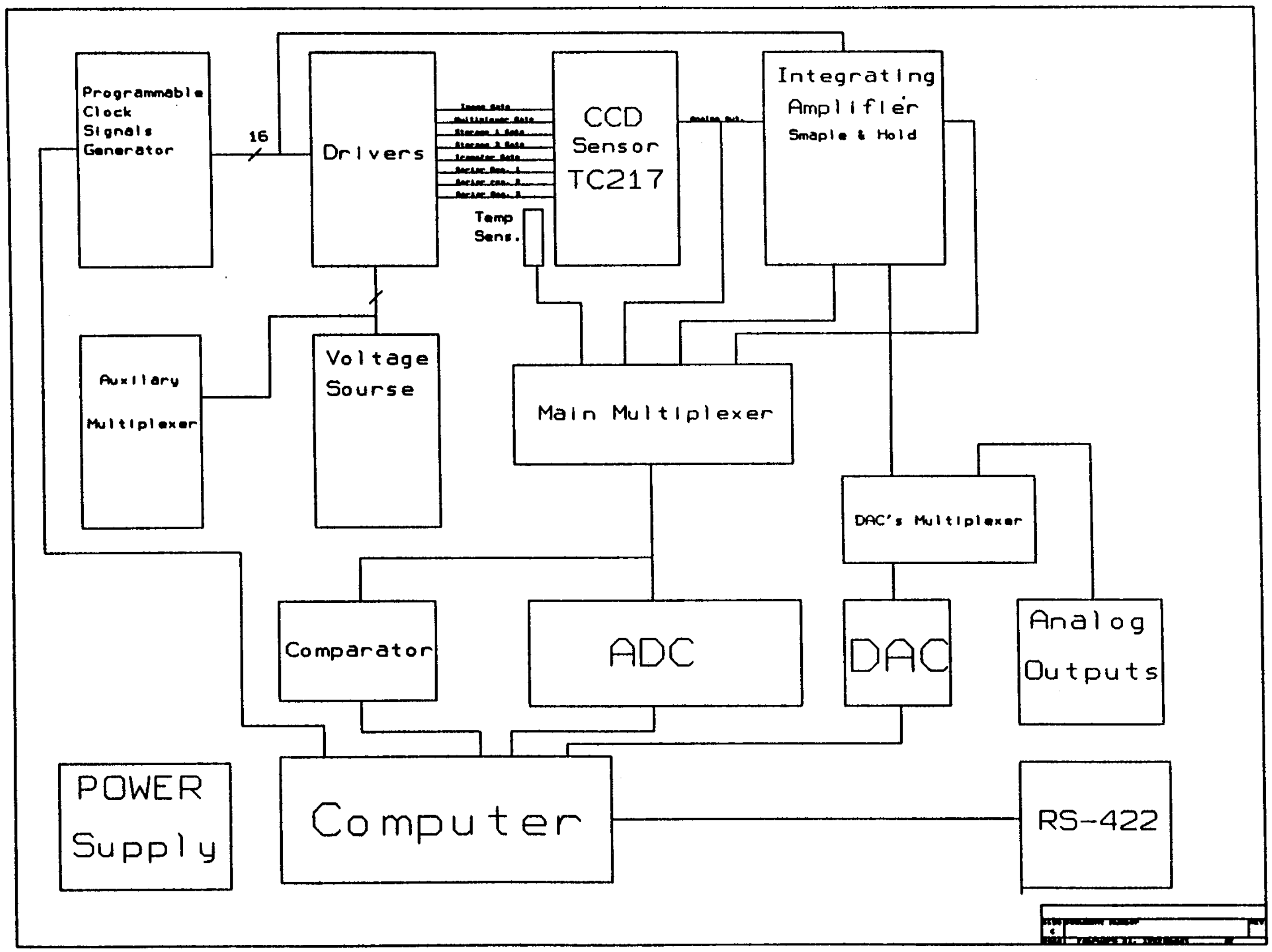






Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

ELECTRONICS



SIGNAL CONDITIONER

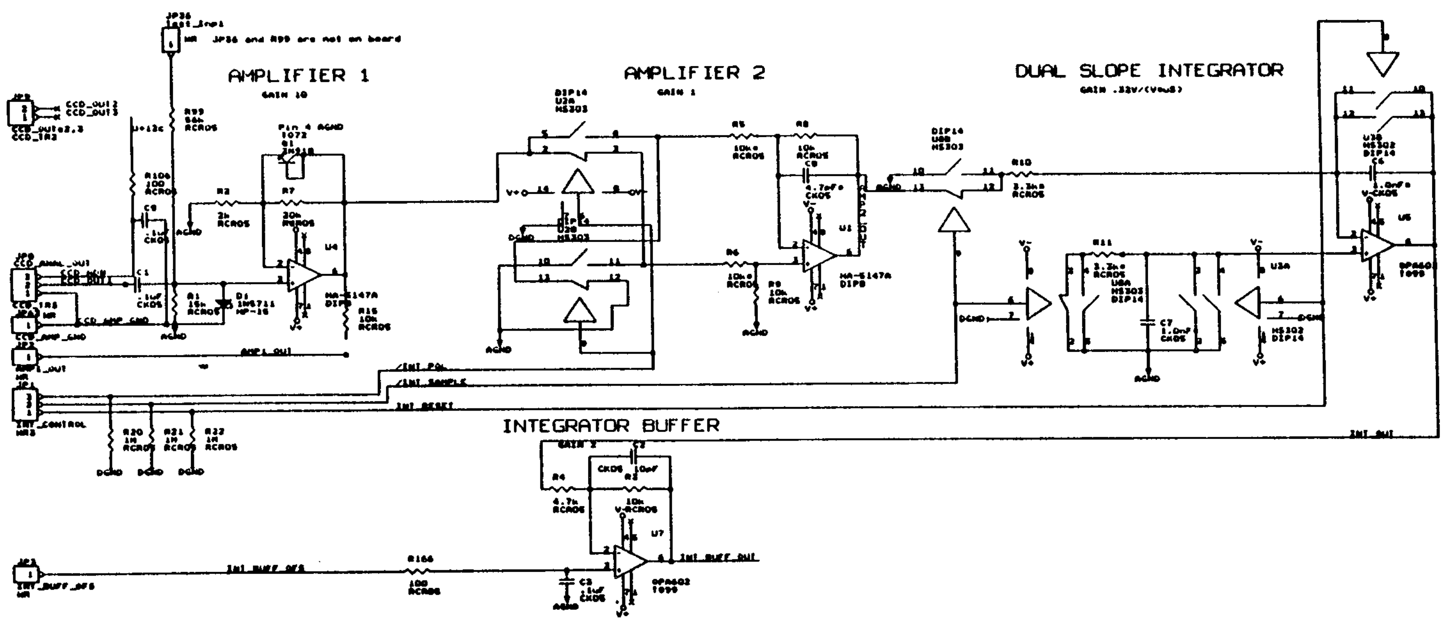
Modification of Kitt Peak design Dual Slope Integrating Amplifier. Relatively complicated but offering best signal to noise ratio.

CCD signal can be monitored at:

- output of CCD
- after first amplifier
- after integrator buffer (normal mode)

Parts:

- Harris HA-5147A Op Amp (fast)
- Burn Brown OPA 602 FET Op Amp
- Harris: HS302, HS303 analog switches



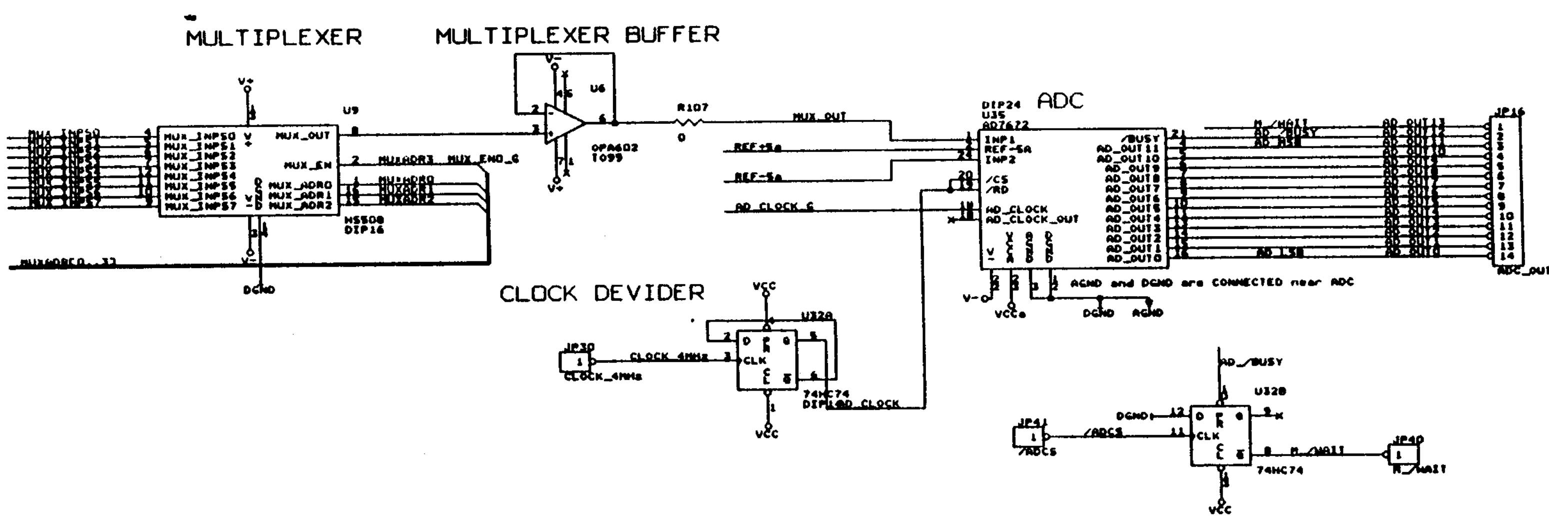
AD CONVERSION

Main Multiplexer: Harris, HS508 8 to 1

Output Buffer: Burr Brown, OPA602 FET OP Amp

ADC: Analog Devices AD7672 12 bit 5µS
Busy signal used to synchronize with CPU

Comparator - used for test search
LM193 -> generates STOP signal to PCSG and interrupt for CPU. Threshold level is programmable through DAC and sample/hold amplifier.



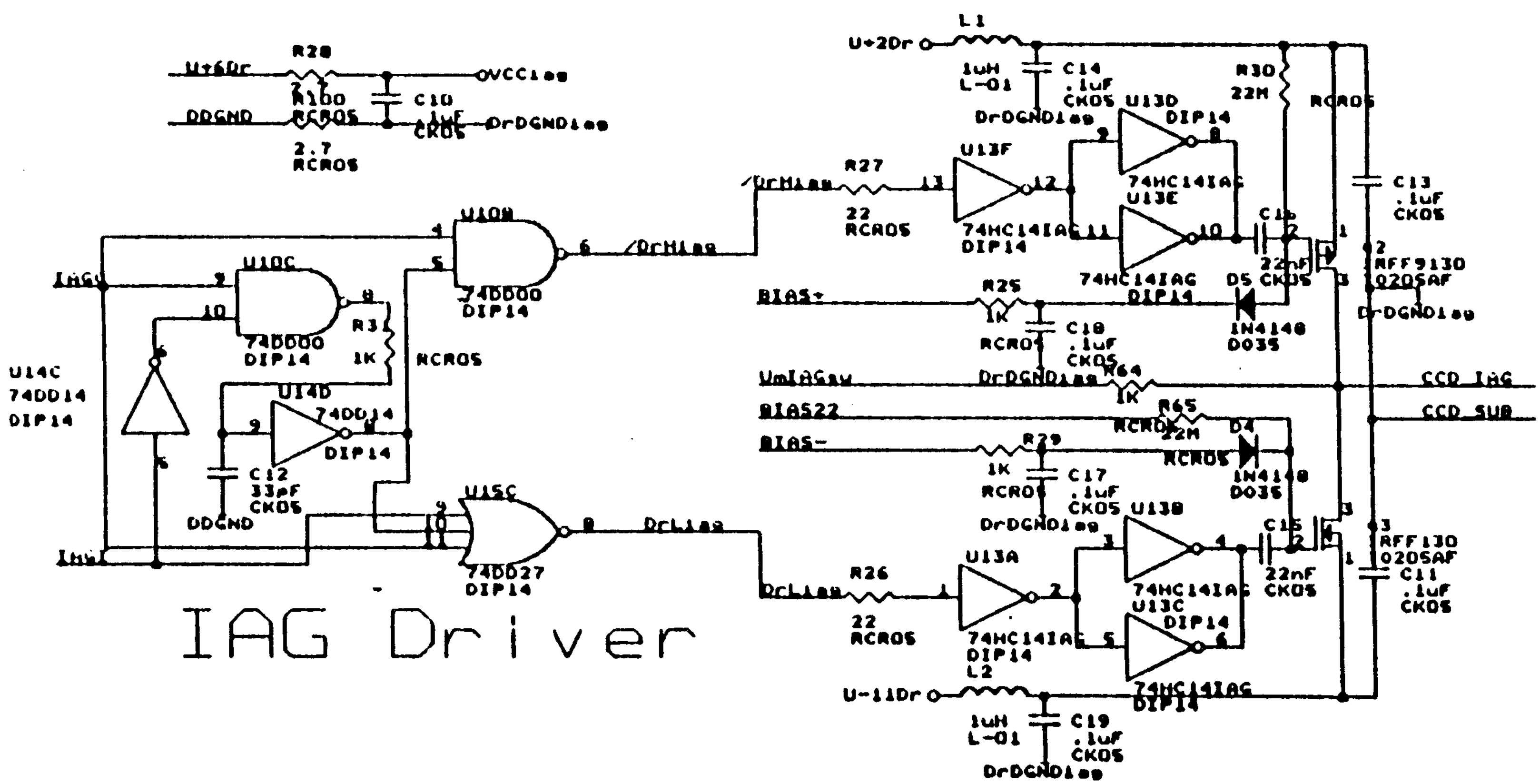
CCD DRIVERS

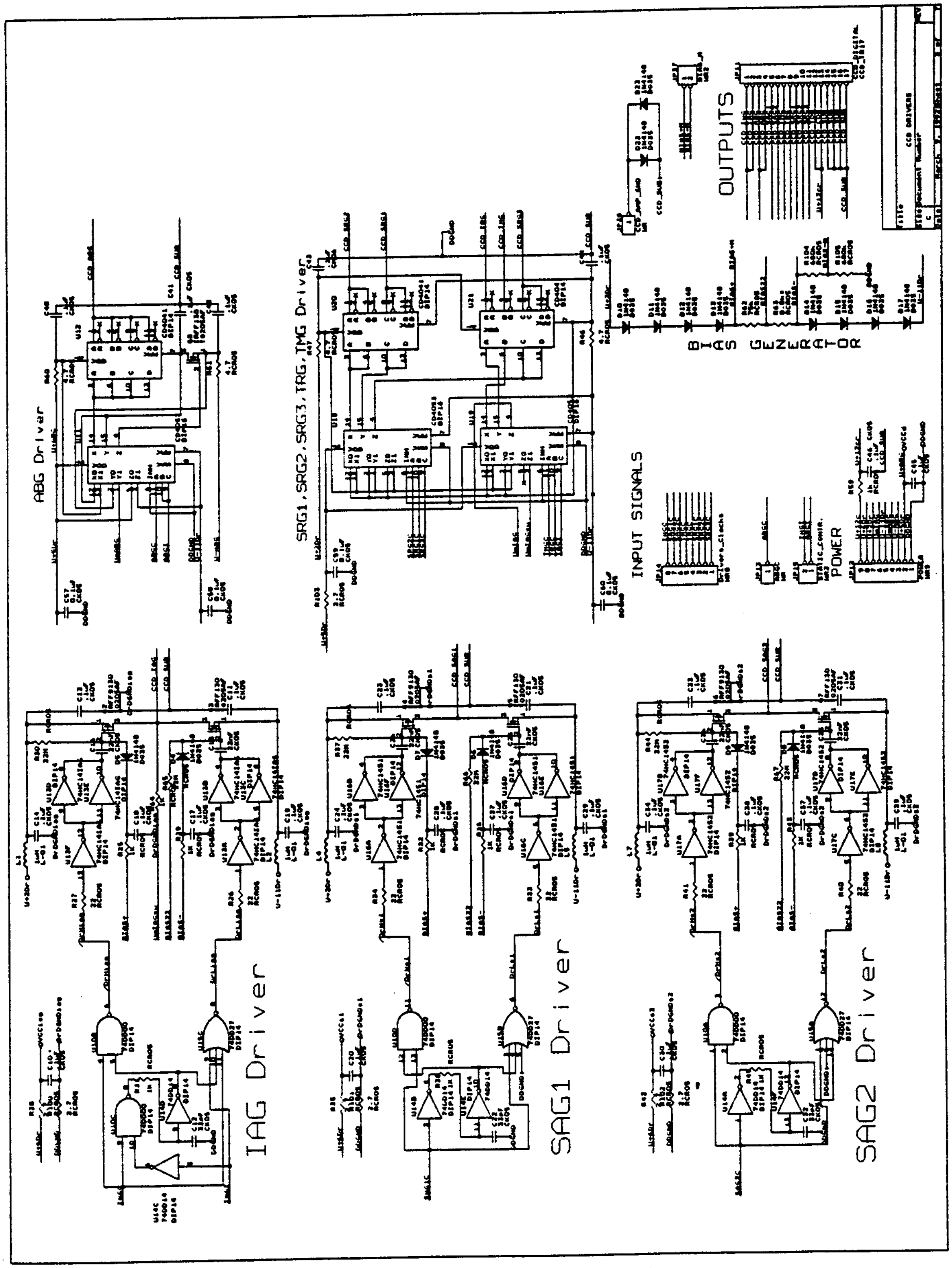
Three CCD gates with capacitance up to 13nF each require power drivers to achieve rise/fall times of 100nS at 12V amplitude (I peak = 1.5A).

Driver is based on rad hard Harris power mosfet transistors FRL9130 & FRL130 (commercial equivalents IRFF9130 & IRFF130).

When not clocked, driver is not using any power.

Remaining gates are clocked by CD4041 buffers.



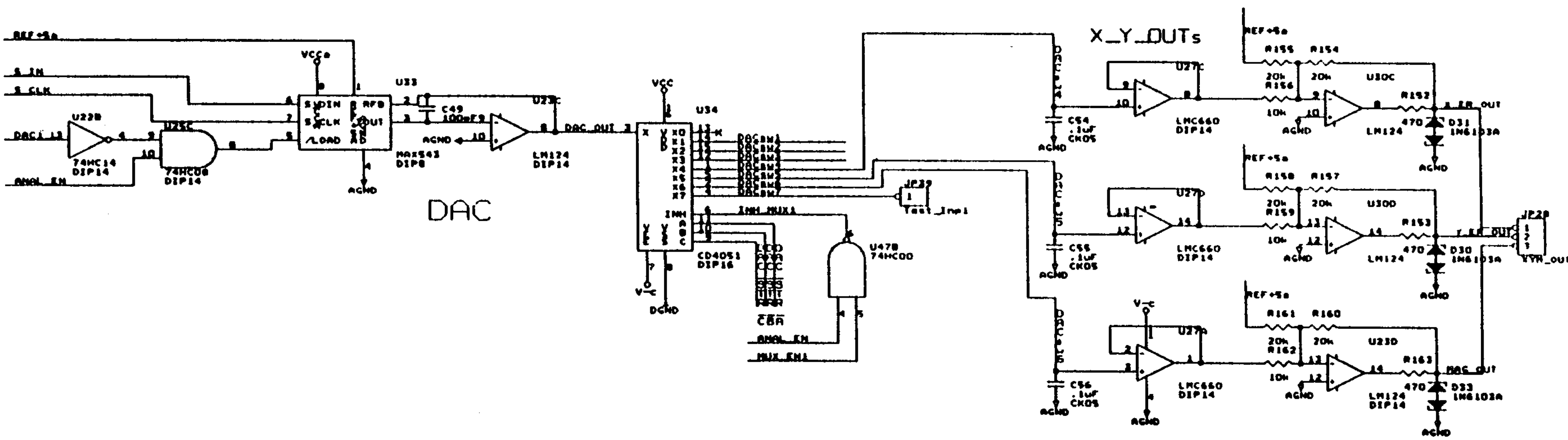


IAG Driver
SAG1 Driver
SAG2 Driver
ABG Driver
SRG1,SRG2,SRG3,TRG,TMG Driver
INPUT SIGNALS
POWER
BIAS GENERATOR
OUTPUTS
CCD DRIVERS

DAC & ANALOG OUTPUTS

To save power and real estate only one DAC was used. We selected MAXIM MAX543 because of serial interface and small package. Output of DAC is multiplexed to several S&H amplifiers.

Overvoltage protection up to 0.5 Joule surge



PROGRAMMABLE CLOCK SIGNAL GENERATOR (PCSG)

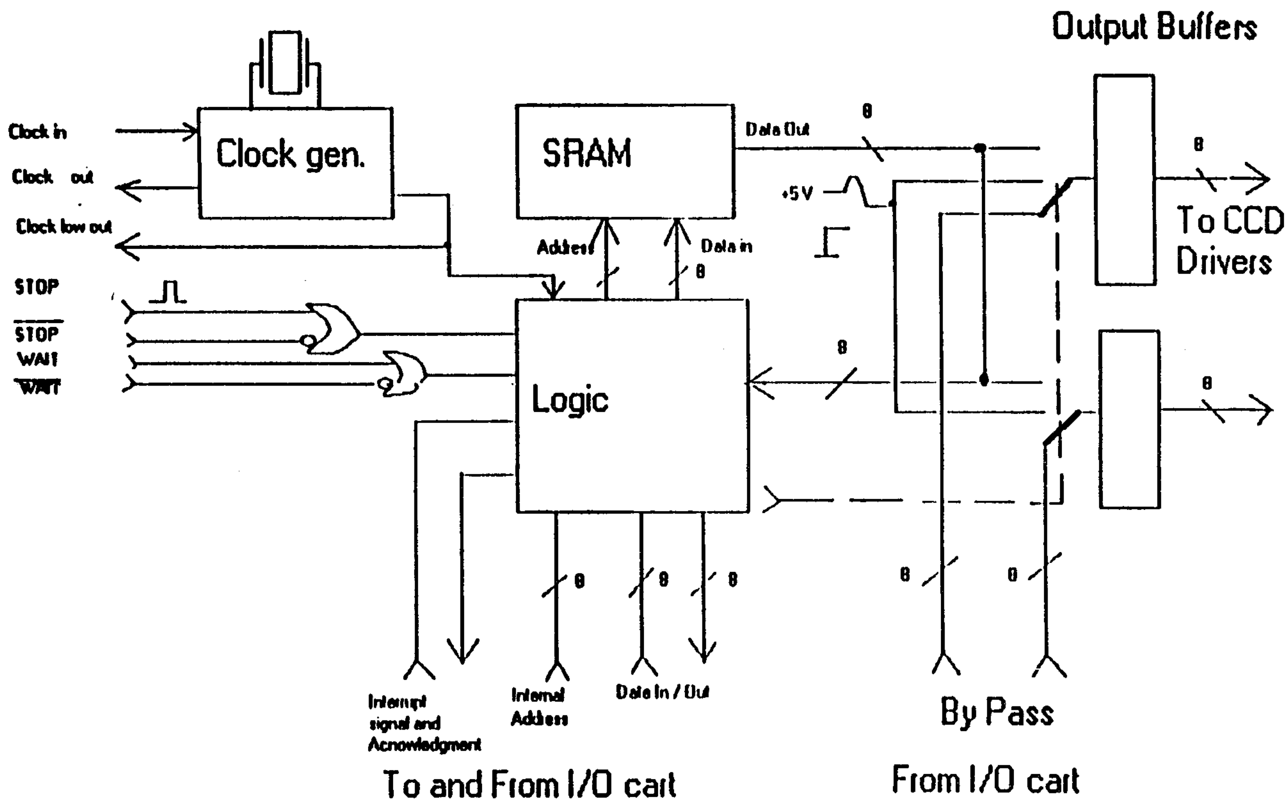
Programmable sequence allows generation of series of clock signals for CCD. Allows fast image transfer to storage area (electronic shutter) and fast clearing of CCD after overexposure.

When programmed and started it operates independently of CPU.

Synchronization with CPU is achieved through interrupts generated by PCSG.

Allows parallel operation:
- CPU calculates star(s) position
- PCSG controls exposure, image transfer and, partially, digitization.

Parts:
- HC series and CMOS
- Memory 2 x OW62256 32K x 8



ON BOARD COMPUTER

CPU:

Harris: HS80C86

Clock:

Harris: HS82C86 12 MHz crystal, allows "sleep" mode

Timer, Baud clock generator, watchdog:

UART: Harris: 82C54

Plessey: MA28151

I/O:

Harris: HS8255 and 54HC573

RAM:

OmniWave: OW62256 128K bytes

EEPOM:

SEEQ: 28C256 128K bytes

Interrupt controller:

National Semiconductors: 82C59

RS-422 full duplex (4 wires):

TxD+
TxD-

RxD+
RxD-

Chassis GND

9600 bauds

UART: Plessey MA28151

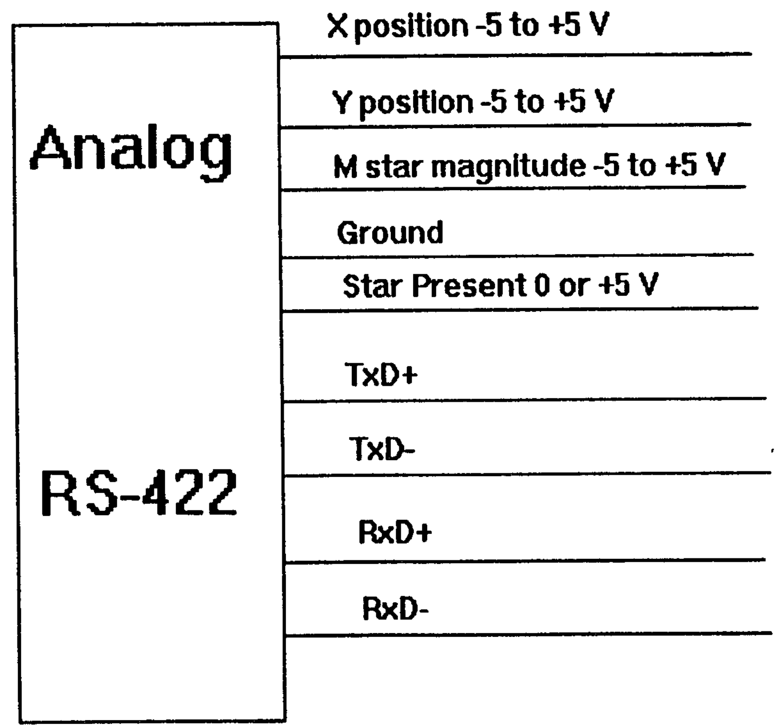
Drivers/receiver:

National: DS26C31MJ/883
DS26C32AMW/883

Overvoltage protection:

Space on boards for surge protection diodes, not tested.

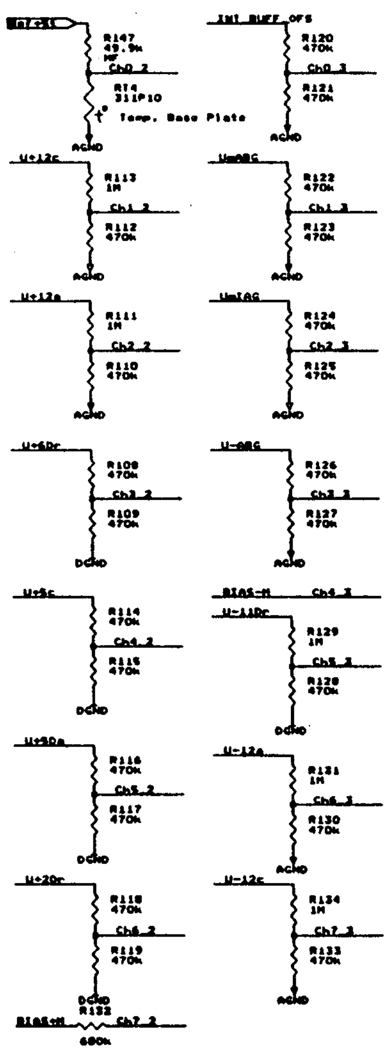
Digital and Analog Interface



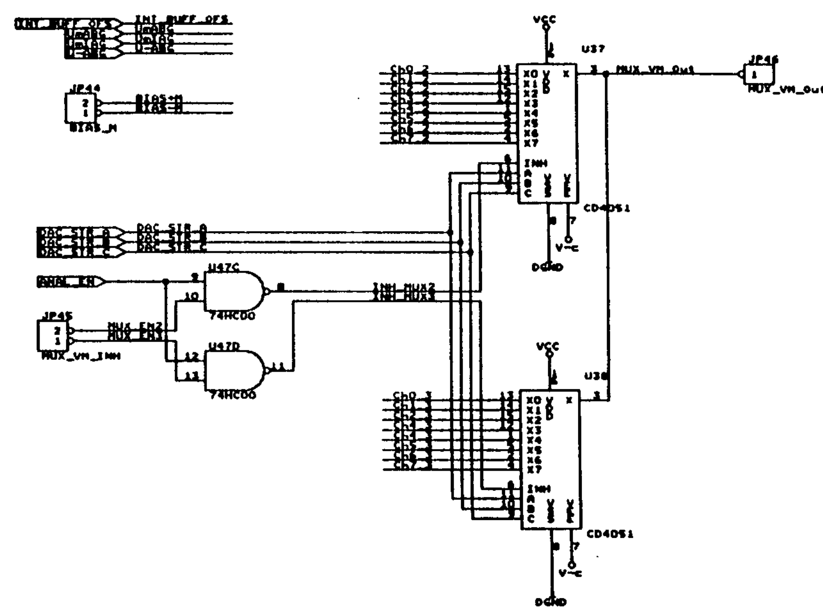
HOUSEKEEPING

Temperature of CCD, lens, thermoelectric cooler and electronic can be monitored by ADC through multiplexer. Temperature of base plate and all internal voltages can be monitored by ADC through auxiliary multiplexer (voltage monitor).

Voltage Deviders



Voltage Monitor



HOUSEKEEPING DATA

Housekeeping data is read during power up of ARC star tracker (there is a 1 minute delay, this allows warm up and self test software to execute). The results are stored in memory and can be read out through a serial port or, in test version of software, displayed on the monitor.

Internal voltages are measured at 24 points, and table entitled Housekeeping Tests summarized for each test point numerical value (as tested with external voltmeter or oscilloscope), typical A/D output, and acceptable error.

The results of self tests have been presented to NASA representatives during program status review.

Test #	Measured Quantity	Nominal Value	A/D Output	Acceptable Error
0	Bias 22m Voltage	-8.1V	385	+/-10%
1	Power +12V Continuous	+12V	3619	+/-10%
2	Power +12V Analog	+12V	3619	+/-10%
3	Power +6V Driver	+5V	3072	+/-5%
4	Power +5V Continuous	+5V	3072	+/-5%
5	Power +5V Analog	+5V	3072	+/-5%
6	Power +2V Driver	+2V	2458	+/-10%
7	Bias+ Voltage	0V	2048	+/-.1V
8	Int. Buffer Offset	-1.5V	1740	+/-10%
9	ABG Voltage Medium	-1.5V	1740	+/-10%
10	IAG Voltage Medium	-5.5V	922	+/-10%
11	ABG Voltage	-7V	615	+/-10%
12	Bias- Voltage	-9.2V	164	+/-5%
13	Power -11V Driver	-11V	607	+/-10%
14	Power -12V Analog	-12V	476	+/-10%
15	Power -12V Continuous	-12V	476	+/-10%
16	Temp. of Volt Ref.	.63V	2306	Only Chip Activity
17	Output of Voltage Monitor	Repeats Value of Selected Channel 0 to 15		
18	Lens Temperature			
19	Int. Buffer Output			
20	Temp. of Peltic Coolers			
21	Direct Output of CCD	+4.9V	4050	In Dark Only
22	Amplifier #1 Output			
23	Temp. of CCD			

REMARKS

1. Temperature channels (#18, #20, and #23) will be calibrated after final assembly.

2. Channels #17, #19, #21, and #22 are used to monitor dynamic properties circuit and output value depends on circuit timing and exposure condition of CCD.

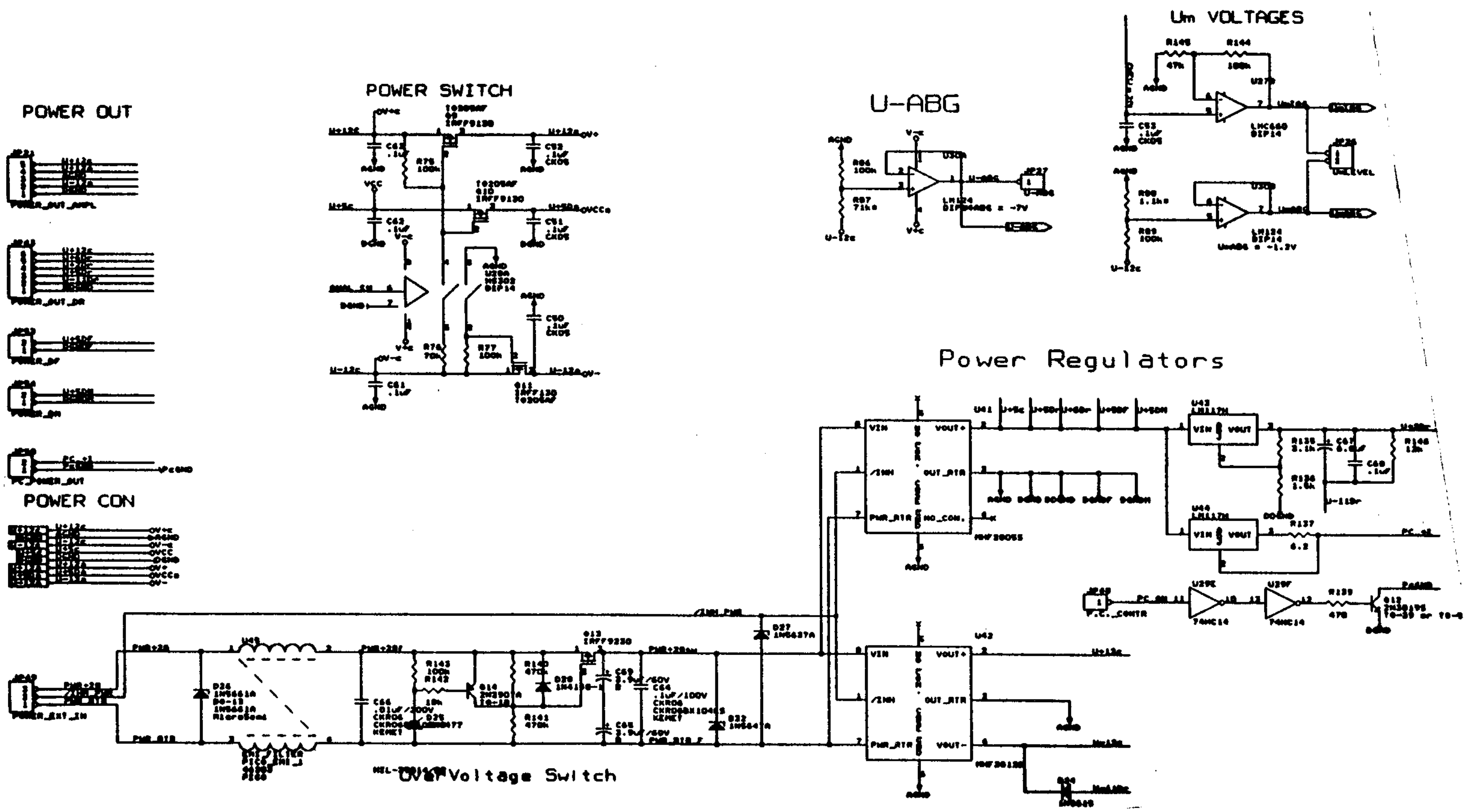
3. Channel 16 is used to monitor if reference voltage generator operates properly. Due to 10M resistor (preventing any significant load) the A/D convertor output indicates much lower voltage (about 0.2).

POWER SUPPLY

Power is supplied by two DC-DC converters Interpoint MHF2805S and MHF2812D capable to deliver 5V/2A and +-12V/500mA respectively. Both converters can be replaced with 883 classified version. Several voltages for CCD driver are then subsequently produced by voltage regulators LM117H and op amps. To save power during exposure time all inactive circuits can be switched off. Turn-on time is 3mS.

Overvoltage protection up to 150V after replacing transistor. Surge protection - possible up to 1.5KJ/components not placed into boards.

Power return insulated from ground - up to 500V



SOFTWARE

Developed and tested on IBM-PC (AT) with Turbo Pascal and Turbo Assembler (Borland). Transfered to Embedded System with Locate and TDREM from Paradigm. This approach was selected to reduce development costs, allow flexibility, simplify modification and allows simultaneous testing of partially built hardware in first stages of project.

Major software blocks:

Acquisition
Position(s) Calculating Block
Communication
Housekeeping, Testing, Power Saving
Failure Prevention - Planned

ACQUISITION

Search Modes:

Fast search - uses threshold detector comparator and determines position of stars above threshold. Search of entire array requires several seconds. Can be decreased to 200ms (50ms with compression 3 x 3 pixels into one) by software improvement.

Slow Search (Mapping Mode):

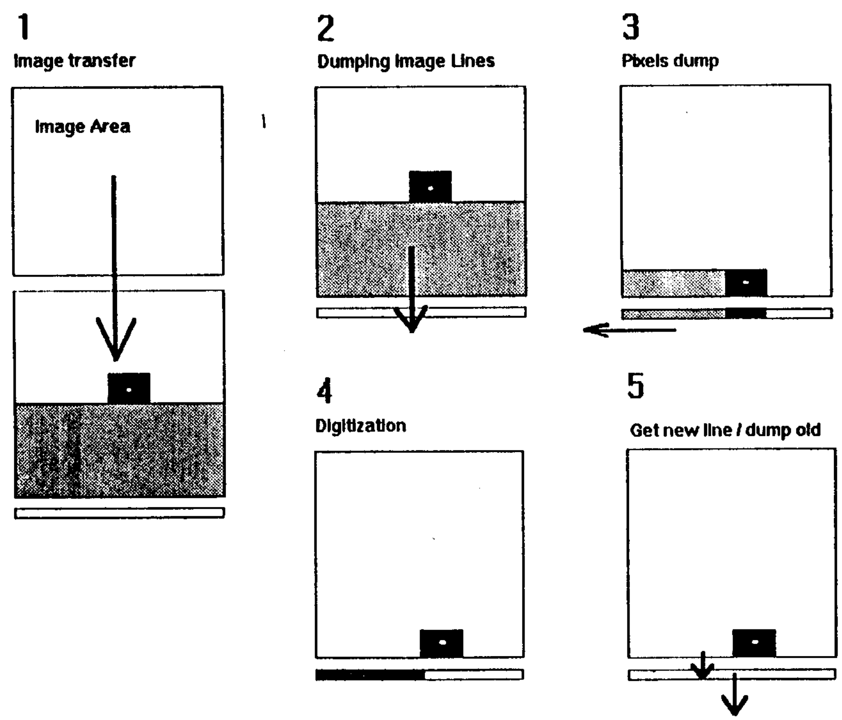
Digitizing every pixel. Appropriate for selective search in selected window. Rate is 20 to 50 times slower than in fast search.

Standard Operation:

- Fast search (approximate exposure time and approximate star position)
- Selective search (position, magnitude)
- Shrink the size of acquisition window around star
- Adjustment of exposure time and fat zero illumination
- Switch to interlaced mode

ACQUISITION IN TRACKING MODE

1. Transfer to image storage area and start exposure - duration .5ms (PCSG).
2. Dump image lines in front of tracking window into clearing drain - duration .2ms (PCSG).
3. Dump of pixels in front of tracking window - duration 0-.3ms (PCSG)
4. Integration and digitizing of about 15 pixels in selected line - duration .15ms (PCSG, CPU).
5. Dump remaining pixels and getting new line - duration .002ms (PCSG).
6. Repeat 3, 4, and 5.
7. a) Wait until exposure is completed then execute 1. and 2. (PCSG); and
   b) In parallel with 7a calculate star position (CPU).



PCSG - task executed by programmable clock signal generator
CPU - task executed by CPU

DECODING AND ADJUSTMENT IN TRACKING MODE

1. TI217 CCD was designed for color TV so each line in shift register contains every third image pixel and next line also contains every third pixel offset by one.

Line 3	3	6	9	12	...
Line 2	2	5	8	11	...
Line 1	1	4	7	10	...

This coding can be easily resolved by CPU. However, it imposes limits on positioning tracking window i.e., it can be relocated only with accuracy of 3 pixels.

2. Fat zero illumination and electrical offset are adjusted based on dark current measurements (monitoring level of dark pixels, dark lines, and pixels just cleared of change).

3. Exposure time adjustment based on maximum intensity pixel.

4. Repositioning tracking window if star position is .55 pixel size out of pixel center (no dynamic repositioning so far).

TRACKING ALGORITHM

Standard mean centroid algorithm was used with modification for background subtraction.

$$X_c = \frac{\sum S_c X_i}{\sum S_c} \qquad S_c = \sum_{columns} (\text{Pixel intensity-background}).$$

$$Y_c = \frac{\sum S_r Y_i}{\sum S_r} \qquad S_r = \sum_{rows} (\text{Pixel intensity-background}).$$

Tracking window was divided into several subwindows:

Star window
Background window
Transition region
Slow-down region

Background is calculated as average in background windows separately for interlaced (shifted) image and for not interlaced (not shifted) image, then subtracted from star window before applying centroid algorithm.

Data from transition region and slow-down region are disregarded. Slow-down region is necessary to stabilize electronic circuitry after changing shift frequency.

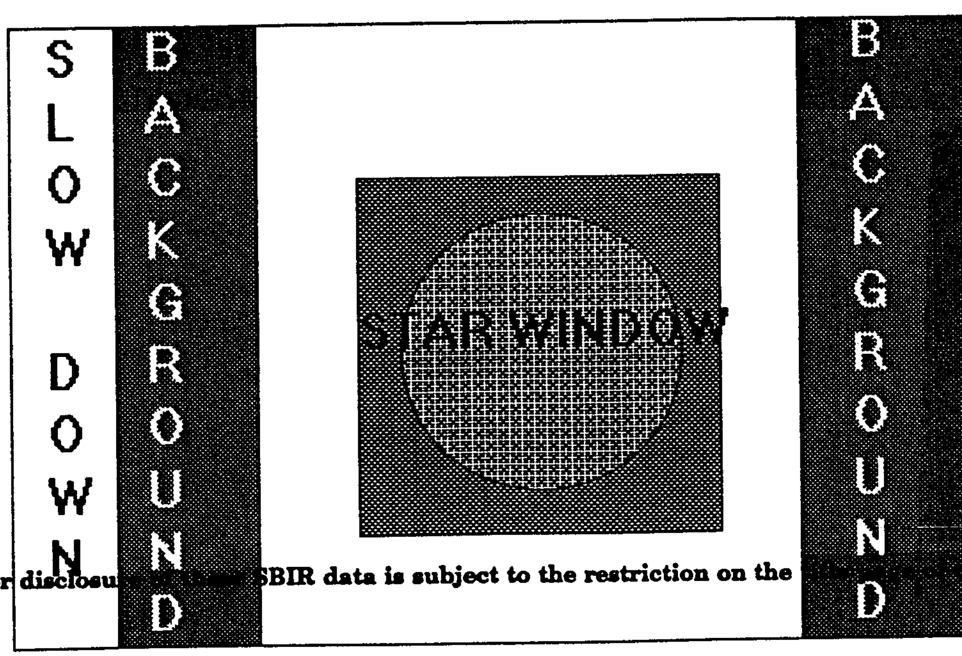
Planned software corrections:

- Position in CCD
- Temperature (CCD, Lens, Base Plate)
- Averaging when changing position of tracking window

Tested but not implemented:

- Fourier corrections on subpixel scale

TRACKING WINDOW



COMMUNICATION

Two types of interface to star tracker are available:

Analog outputs x, y, m, star presence (+5V)
Digital RS-422 bidirectional

After applying power, star tracker is entering test mode followed by search mode. If search successful, star presence signal and x, y, m, outputs are updated. Digital interface is in listen-only mode.

9600 baud transmission rate and response time less than 100ms will be supported by interrupted driven interface to UART. Two implementations of communication interface are considered.

1. Simple (custom written) will allow byte transmission without handshaking. If is sufficient for star position reporting, magnitude and status reporting.

2. Full feature communication package based on Blaise Asynch Plus 5.00 communication routines ported to embedded system will allow large quantities of data to be transferred to and from star tracker. Necessary if program will have to be changed remotely and for full diagnostic (memory and registers readout).

HOUSEKEEPING

Temperatures - measured every frame:

- Lenses
- Base plate
- CCD
- Electronics
- Thermoelectric cooler

Internal voltages (measured on power-up or on request):

- Voltages supply - 10
- Voltages bias - 2
- Voltages reference - 3

IN-FLIGHT TESTING

- Voltage monitoring
- CCD response to LED (three colors)
- CCD test pattern (dark pixels, cleared pixels)
- Monitoring CCD output, of first amplifier
- Telemetric readout of memory

REMOTE MODIFICATION OF SOFTWARE

Communication Kernel is <u>hardware</u> protected from overwriting. Kernel can be modified by loading secondary copy and rebooting system. Rebooting using secondary copy can be then executed automatically by primary (nonmodifiable) kernel on every power up. Tracker main software and secondary kernel is hardware protected for overwriting with possibility to overwrite protection using coded switch. This software can be modified remotely in flight.

PERFORMANCE AND TEST RESULTS

OPTIC SYSTEM

Tested Nikon 50mm/f1.2

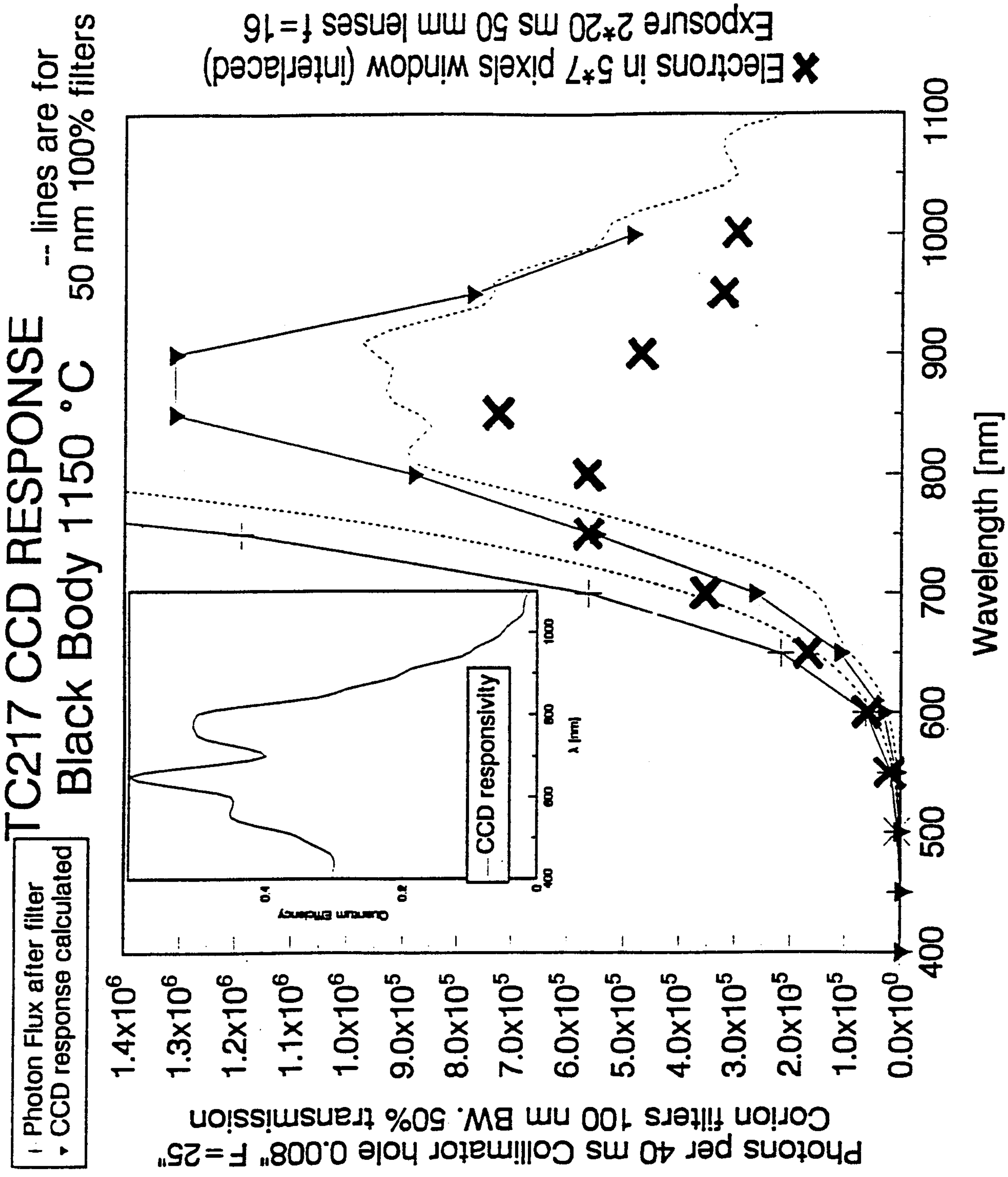
Partially Tested Tuscon 50.3/f1.4

PERFORMANCE OF CCD

Sensitivity

Spectral response of CCD sensor was tested with black body source and Nikon 50mm/1.2 camera lenses. Reasonable agreement with spectral data was found in visible region. In infrared region sensitivity was about 50% of expected.

This discrepancy was attributed to reflection from lens coating in infrared. Tests with new optics are necessary.



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

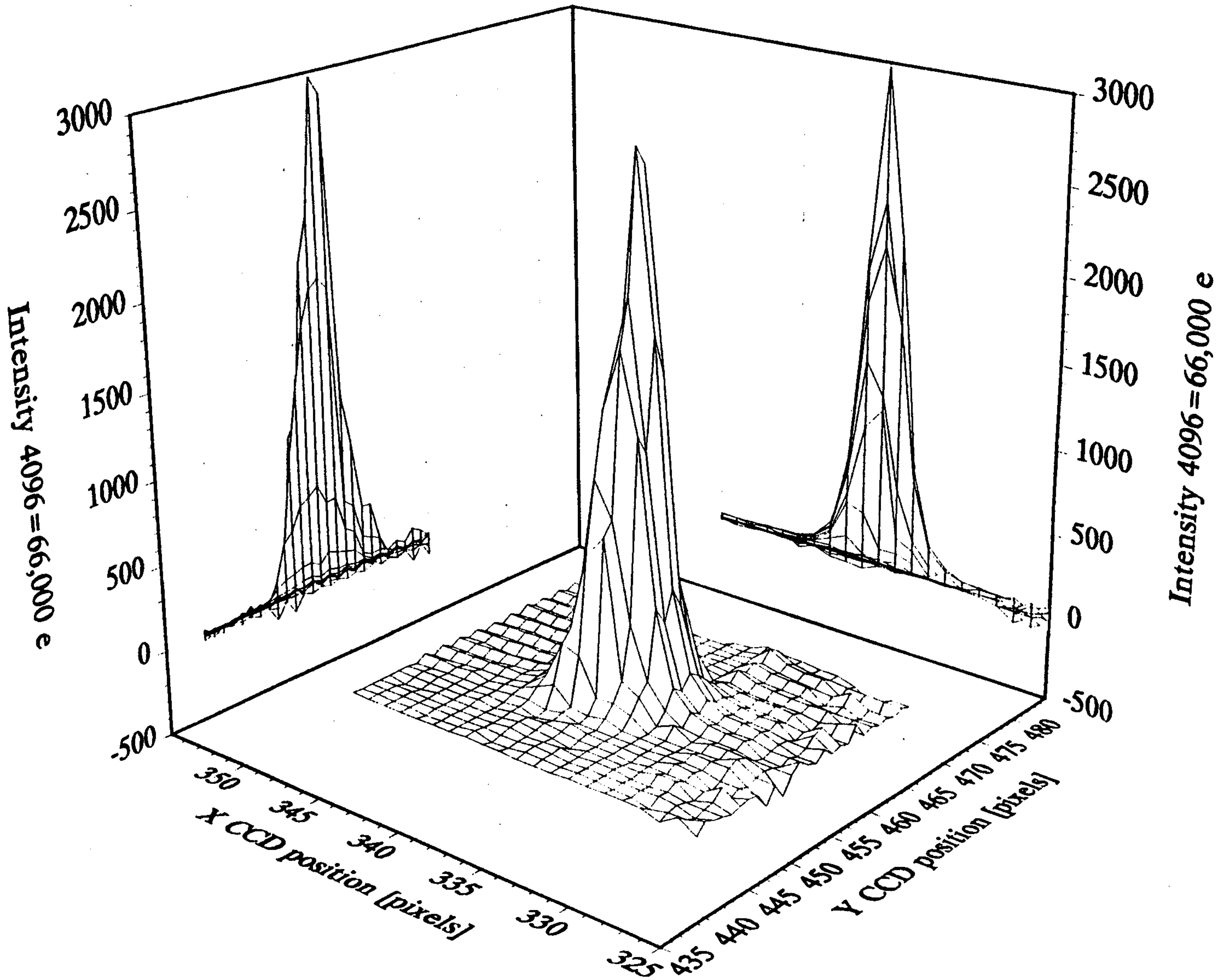
SPOT SIZE

Nikon lenses produce relatively large point spot
(8 x 11 at f = 16, 12 x 18 at f = 1.2).

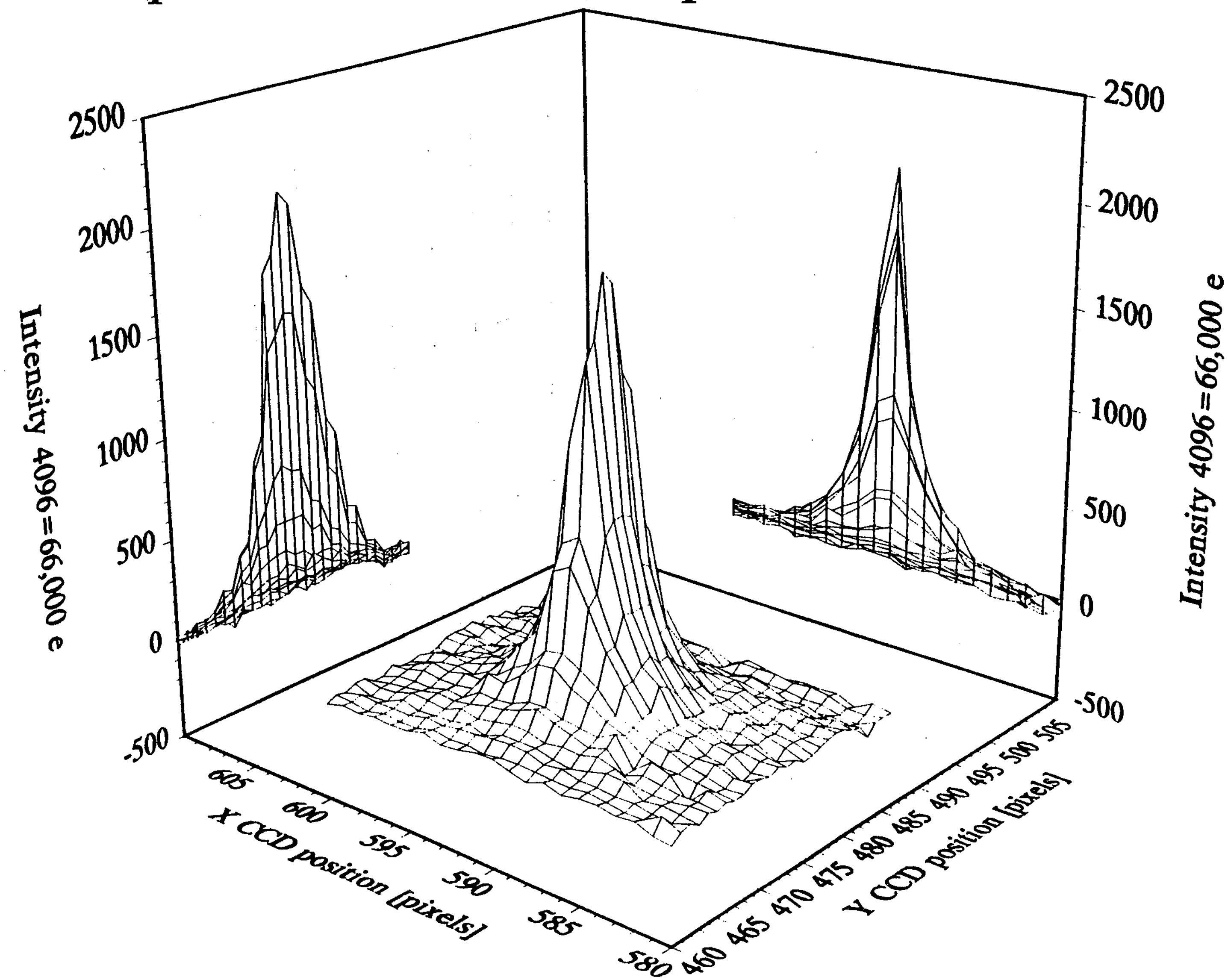
Custom designed lens will produce point spot better matching CCD pixel size thus allows reduction of tracking window size, speeds acquisition, and increases sensitivity by concentrating entire photon flux in smaller number of pixels.

Until after testing with new lenses we can present only estimated performance.

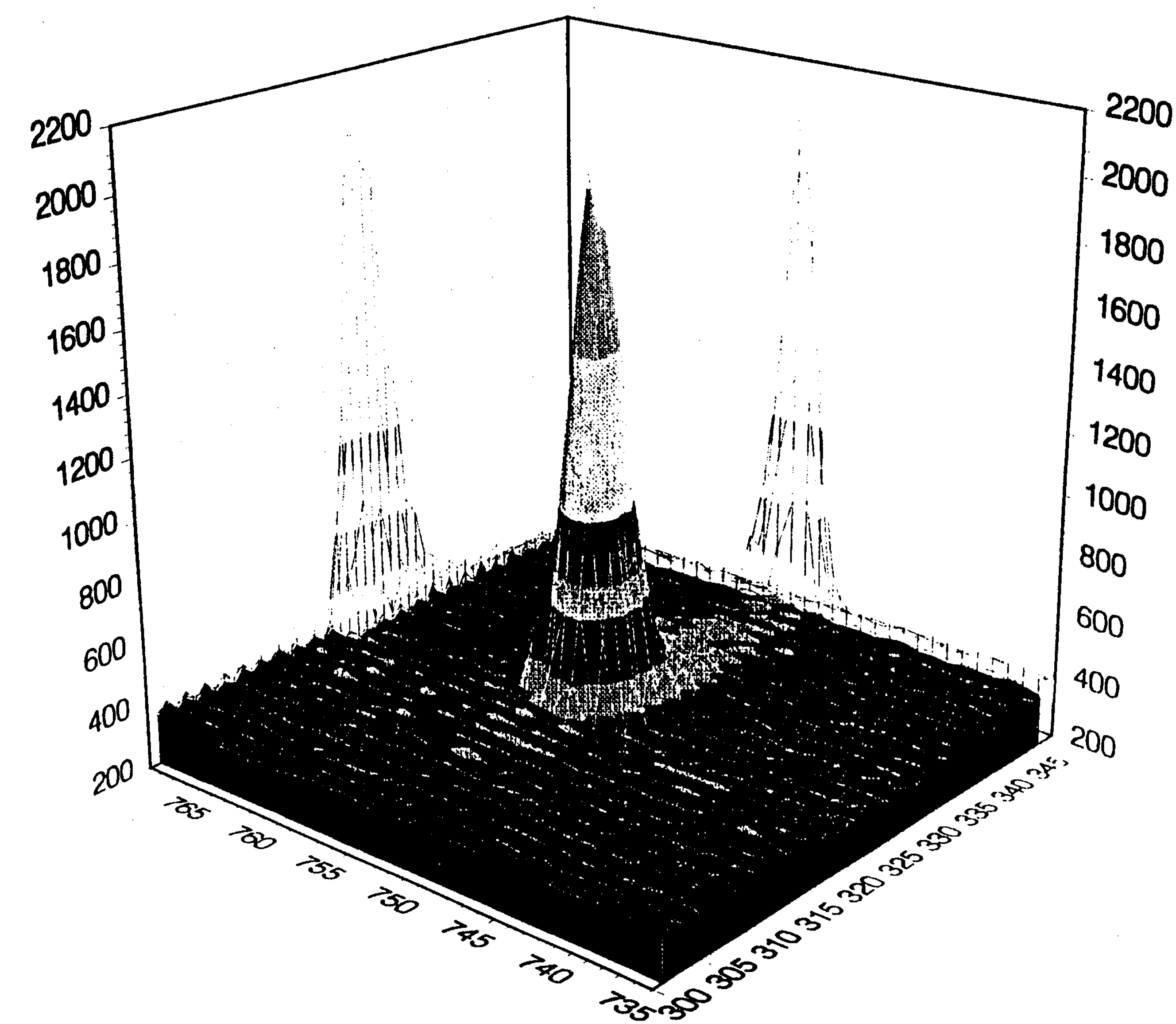
Intensity Profile ∅Hole=20μm
F colimator=635mm; Lens 50mm f=16
X pixel 7.8μm=3.2"; Y pixel 6.8μm=2.8"



Intensity profile ϕHole=20μm
F colimator=635mm; Lens 50mm f=1.2
X pixel 7.8μm=3.2"; Y pixel 6.8μm=2.8"



BB 1010 C H .333*10-3 L=50mm f=5.6
T=20 ms Int=4 us No filter Light ON StInf1



CALCULATED SENSITIVITY

Lens 50mm F/1.4 A = 10cm²
Flux $N = A \cdot 10^{-4m+6.7}$
QDE = 40% Defocusing factor 15% Absorption 15% => 5% in 1 pixel

M	N/Sec	In 1 pix/sec	Exposure Time/Filling of Well (Full Well = 66000 electrons) 1s	100ms	10ms	1ms
0	$50 \cdot 10^6$	$2.5 \cdot 10^6$			38%	4%
1	$20 \cdot 10^6$	$1.2 \cdot 10^6$			18%	2%
2	$8 \cdot 10^6$	$400 \cdot 10^3$		60%	6%	.5%
3	$3 \cdot 10^6$	$150 \cdot 10^3$		23%	2%	
4	$12 \cdot 10^6$	$60 \cdot 10^3$	100%	10%	1%	
5	$.5 \cdot 10^6$	$25 \cdot 10^3$	38%	4%	.4%	
6	$200 \cdot 10^3$	$12 \cdot 10^3$	18%	2%		
7	$80 \cdot 10^3$	$4 \cdot 10^3$	6%	.5%		
8	$30 \cdot 10^3$	$1.5 \cdot 10^3$	2%	.2%		
9	$12 \cdot 10^3$	$.6 \cdot 10^3$	1%			
10	$5 \cdot 10^3$	$.2 \cdot 10^3$	.3%			

Operating Range Room Temperature

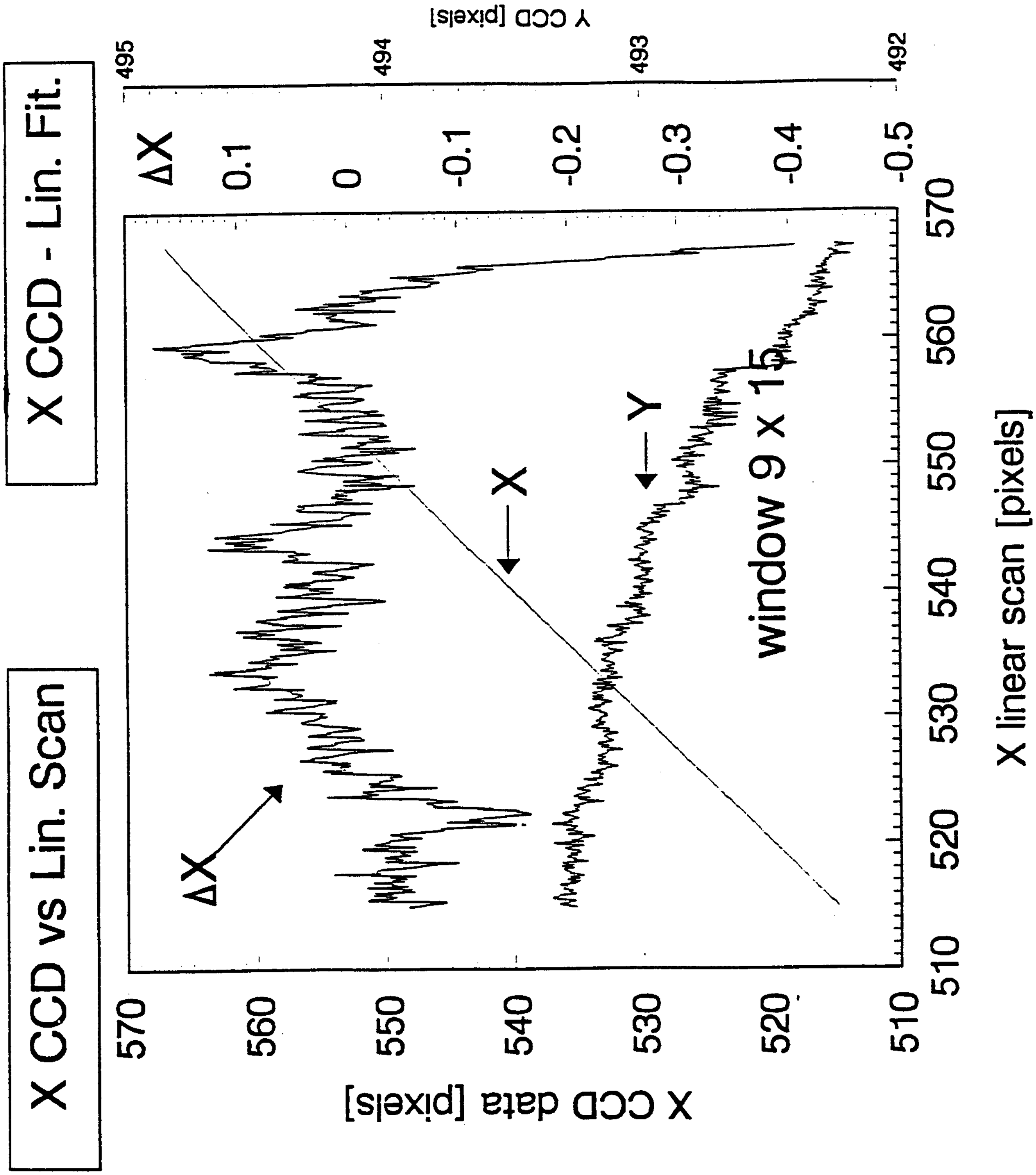
Operating Range with Cooling

LINEARITY

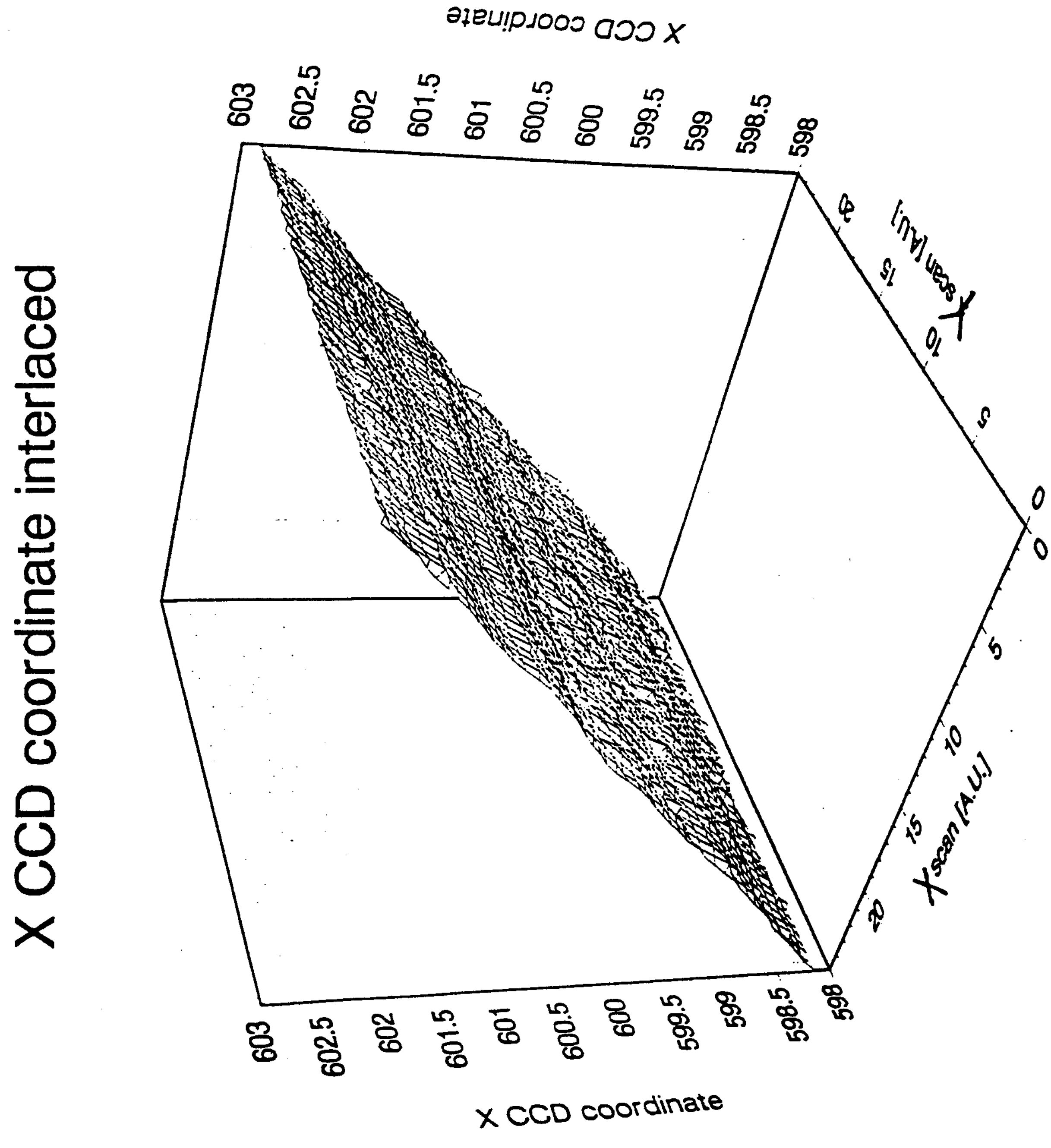
Tracking performance (both static and dynamic) of TC217 CCD was tested with Nikon lenses.

Point spot produced by these lenses is relatively large (about 8 x 11 pixels) even with partially closed diaphragm. So, tracking window 9 x 15 pixels was necessary to achieve linearity about .1 pixel.

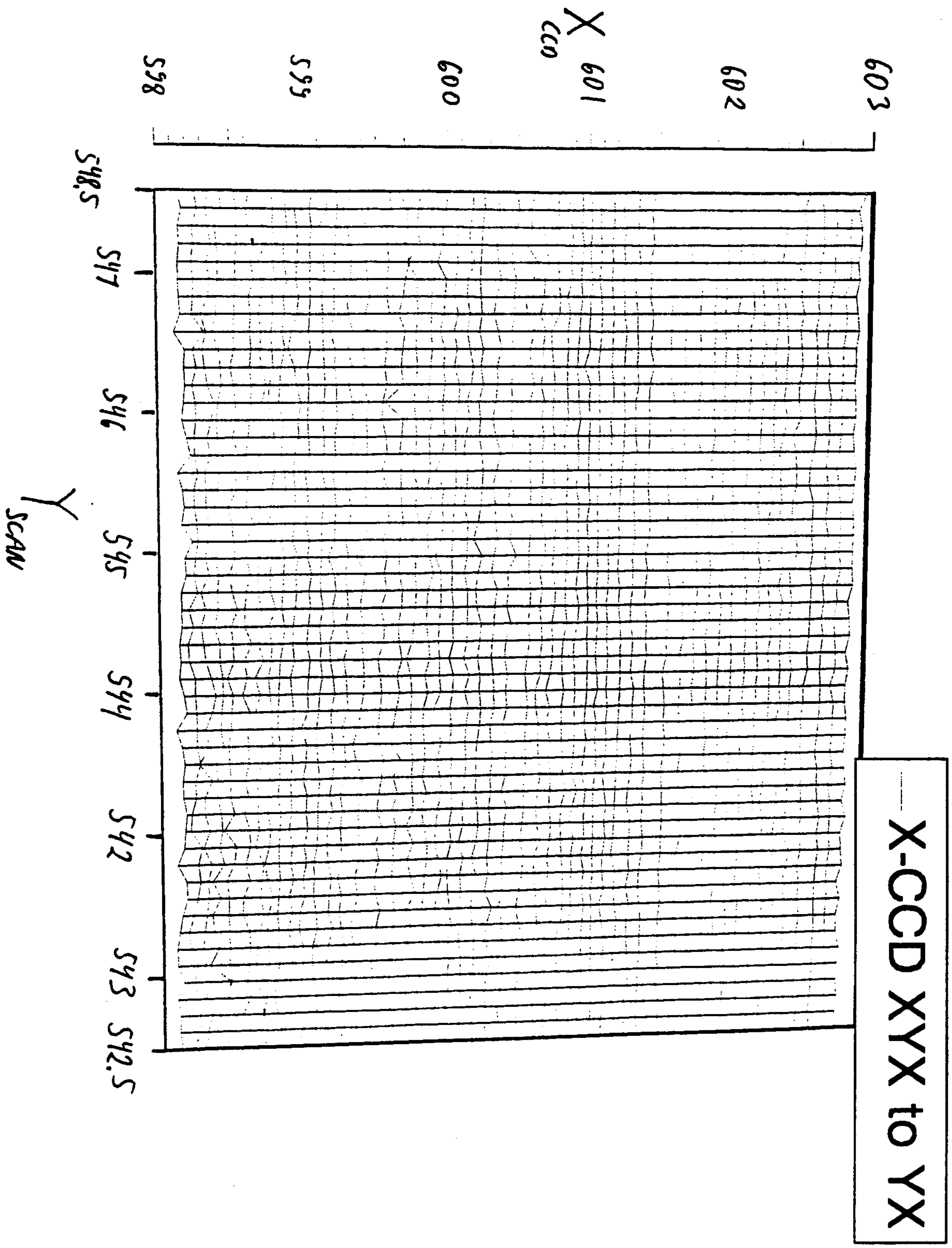
Errors of about .2 pixel are introduced when tracking window was relocated in perpendicular direction. This error can be reduced by calculating position based on relocated and non relocated window and averaging.

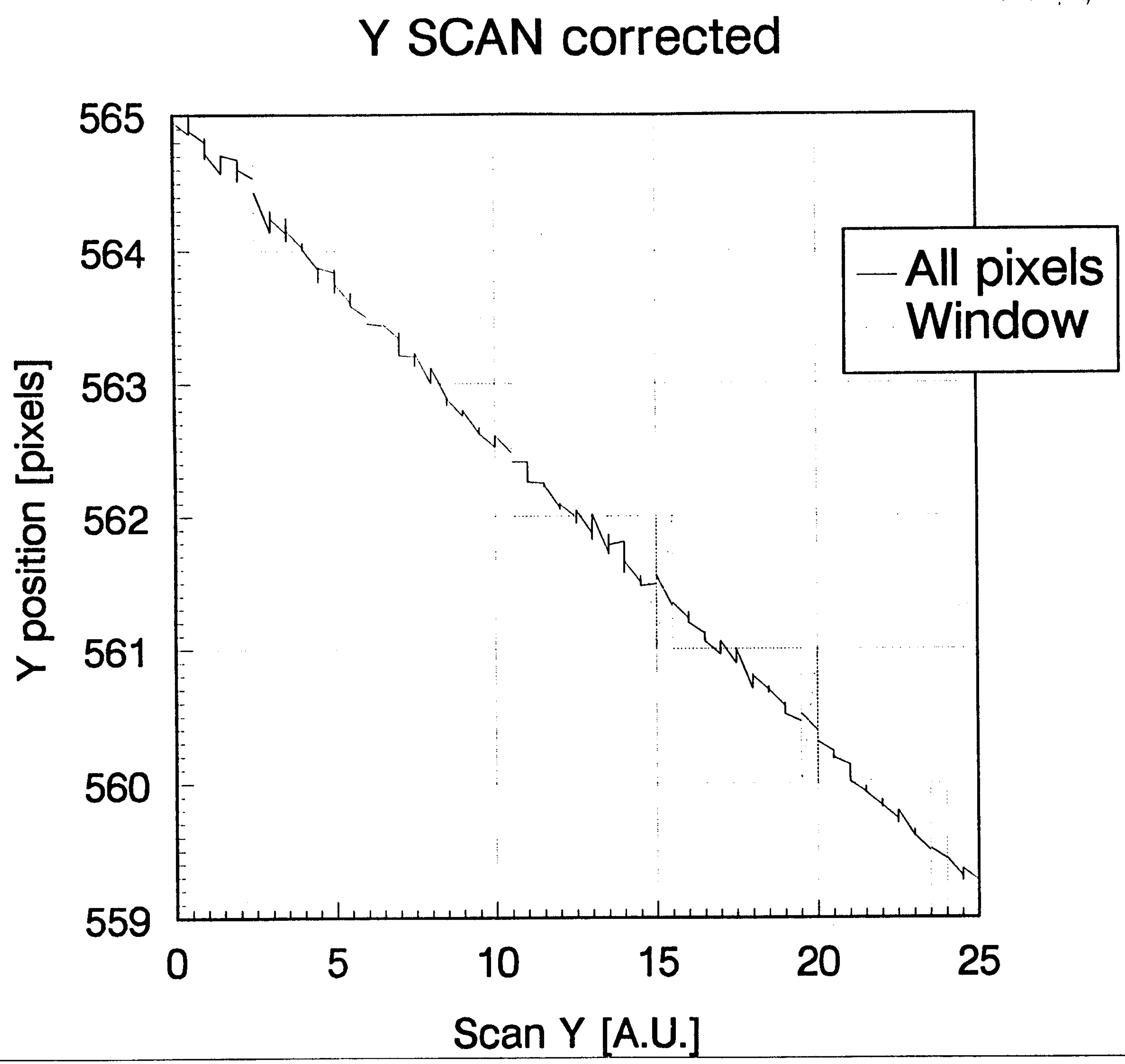


X CCD vs Lin. Scan
X CCD - Lin. Fit.
X CCD data [pixels]
570
560
550
540
530
520
510
X linear scan [pixels]
510
520
530
540
550
560
570
ΔX
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Y CCD [pixels]
495
494
493
492
ΔX
X
Y
window 9 x 15

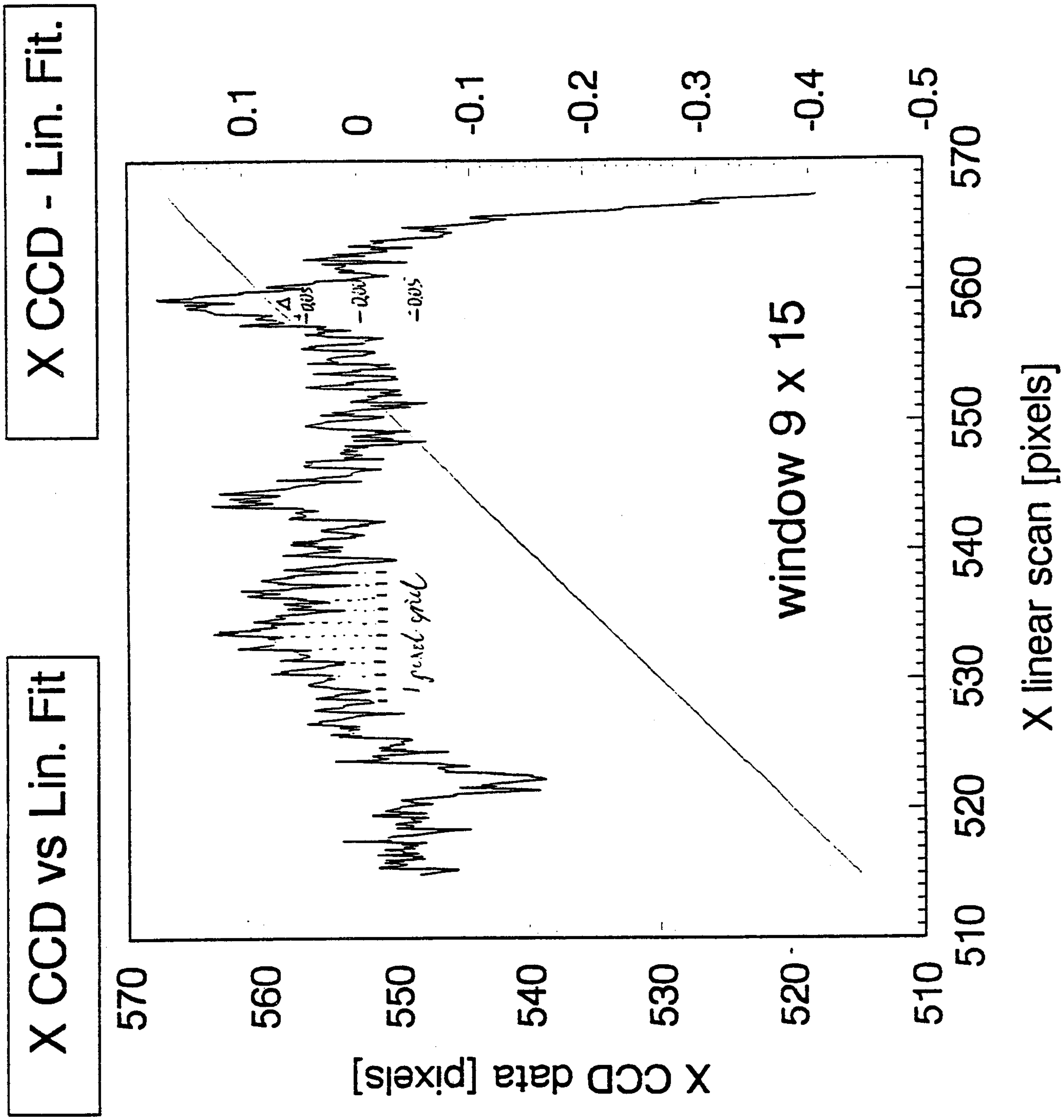


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

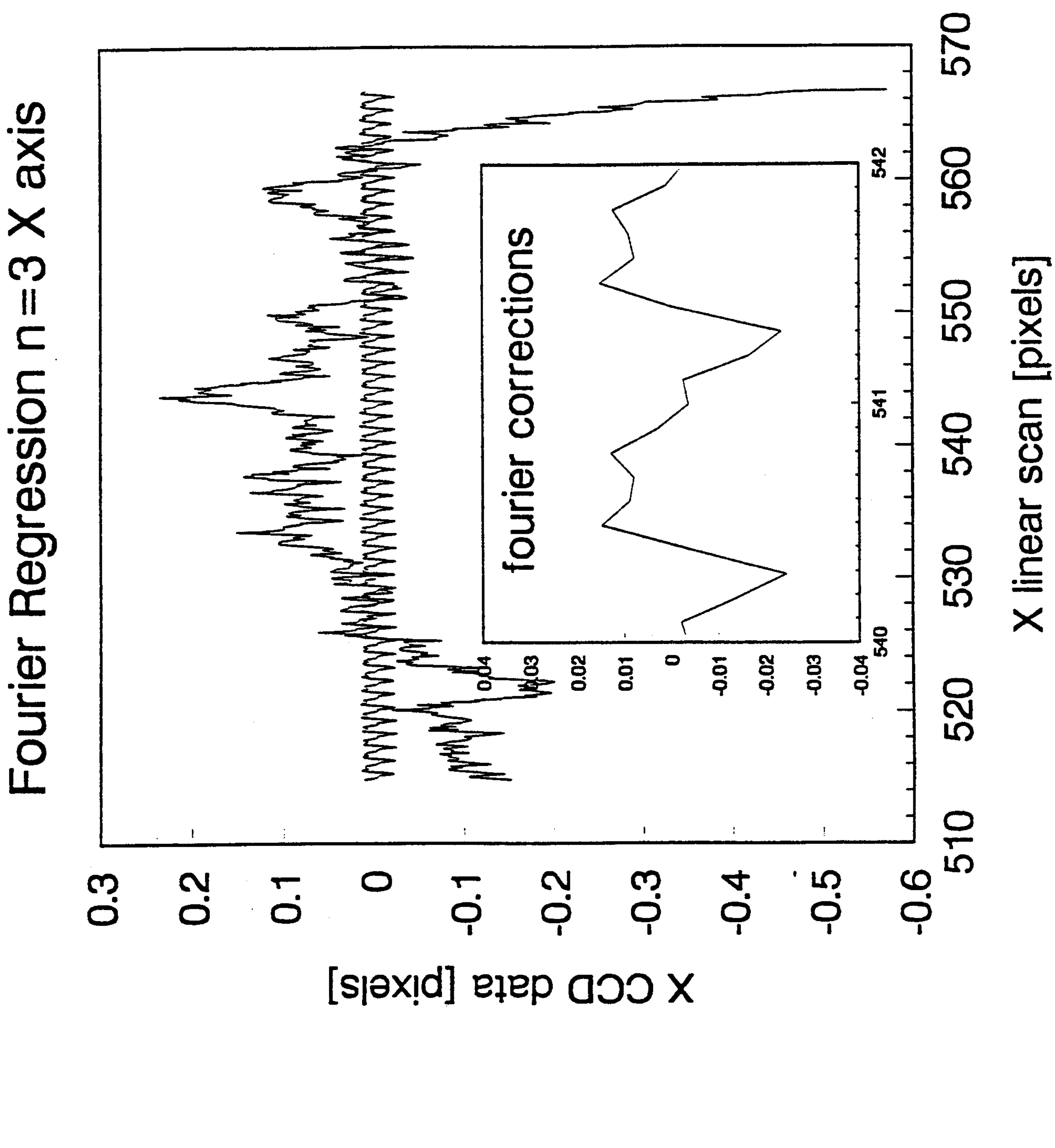

X CCD coordinate interlaced
X-CCD XYX to YX



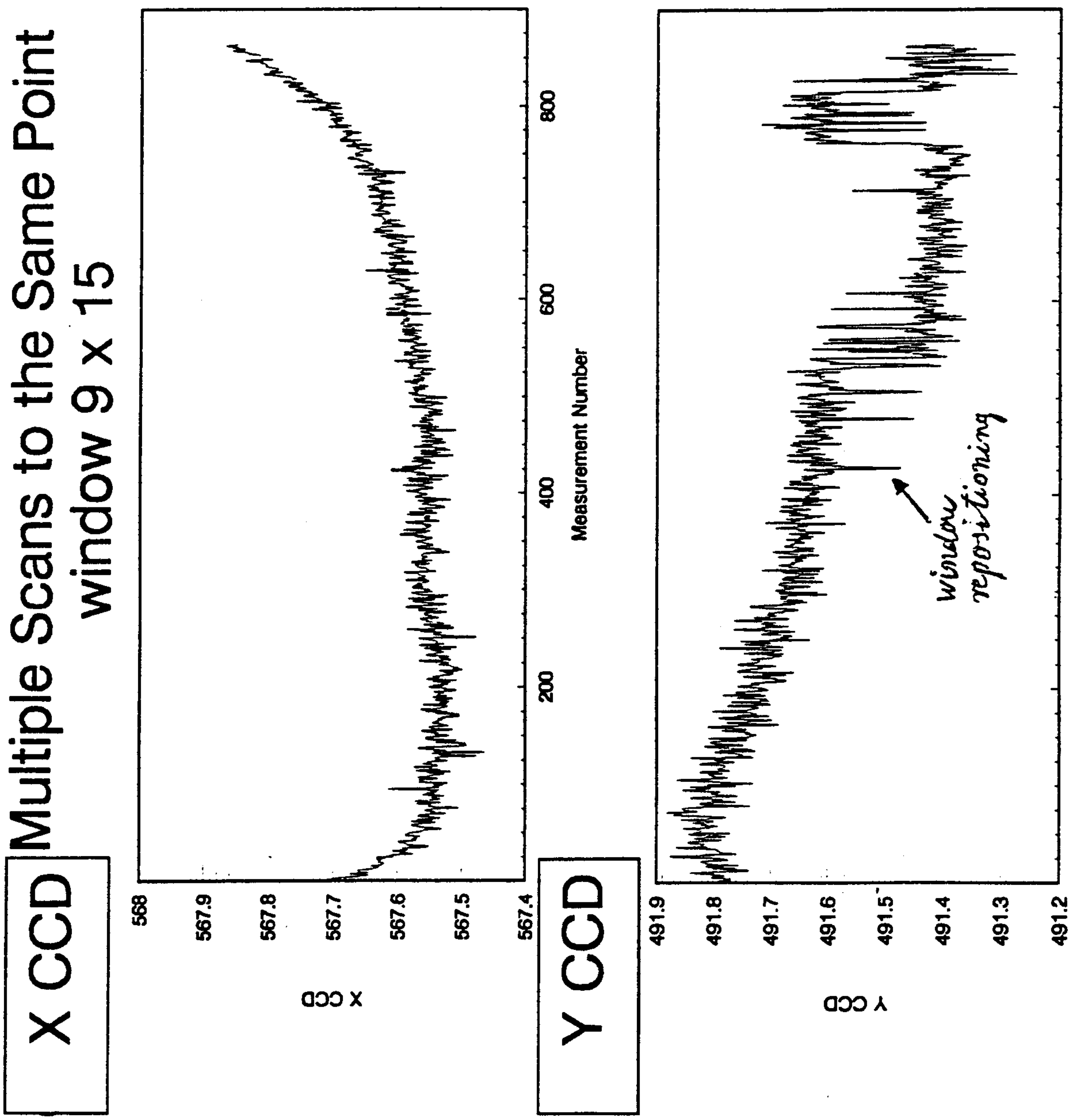
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

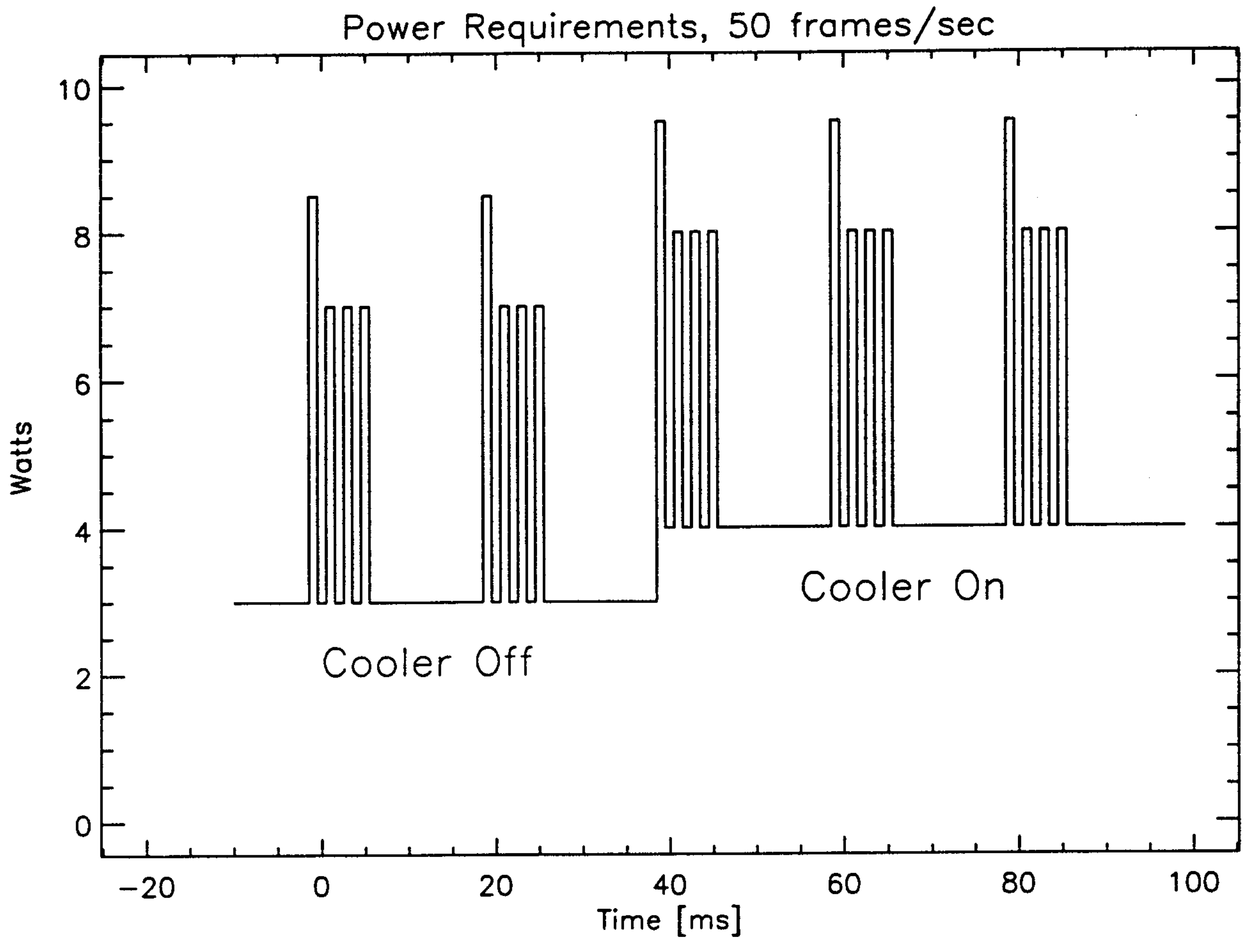


Multiple Scans to the Same Point
window 9 x 15
X CCD
X CCD
568
567.9
567.8
567.7
567.6
567.5
567.4
0
200
400
600
800
Measurement Number
Y CCD
Y CCD
491.9
491.8
491.7
491.6
491.5
491.4
491.3
491.2
window repositioning
Measurement Number

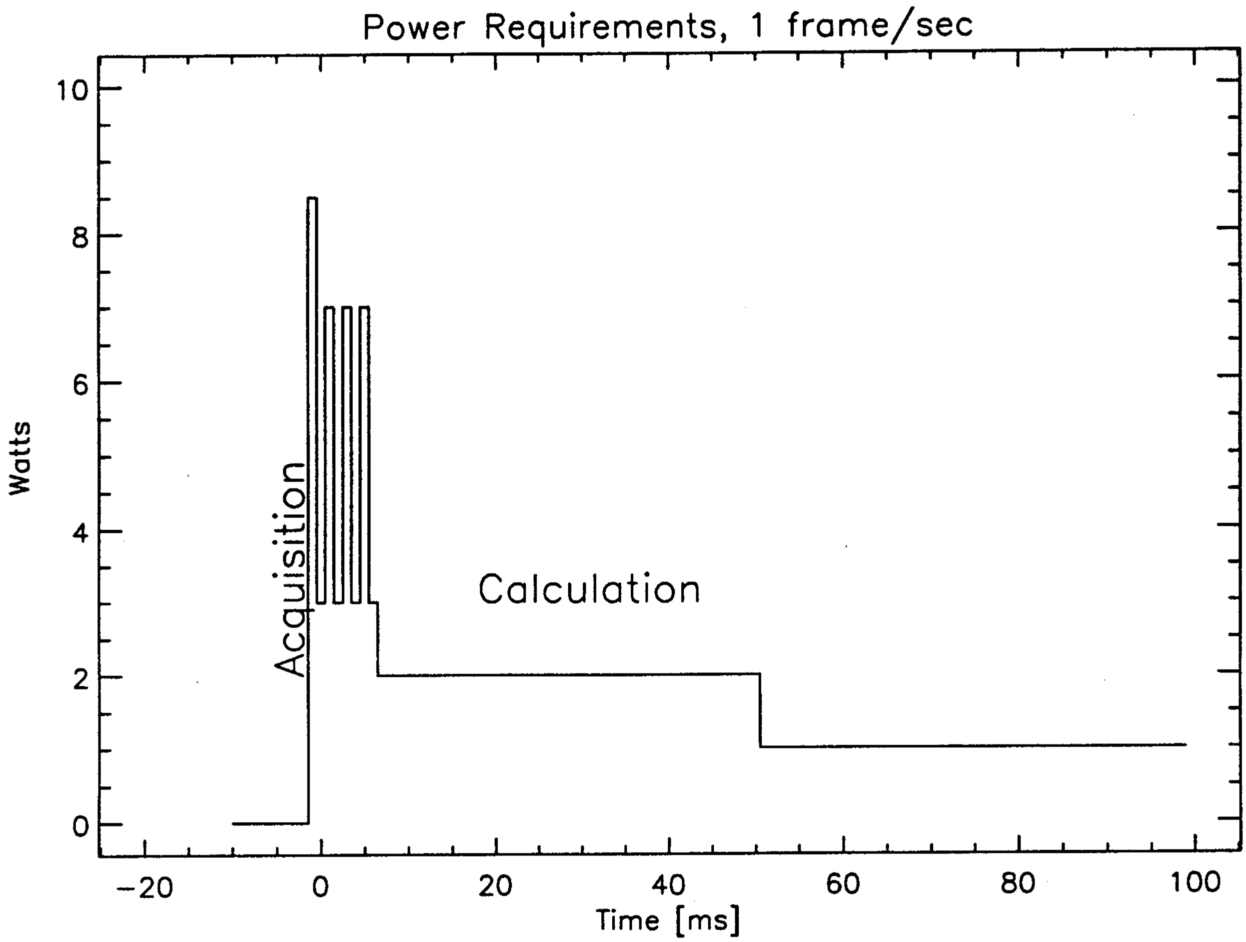
POWER REQUIREMENTS

The main power consumers are CCD drivers and computer. However, CCD drivers do not require any power when not clocking CCD and CPU can be placed in sleep mode after finishing calculation.

DC-DC converters are inefficient on low power load so idle power consumption will be about 1W.



Average power consumption was measured. Subsecond power requirement was calculated on the basis of power measurements for each subsystem.



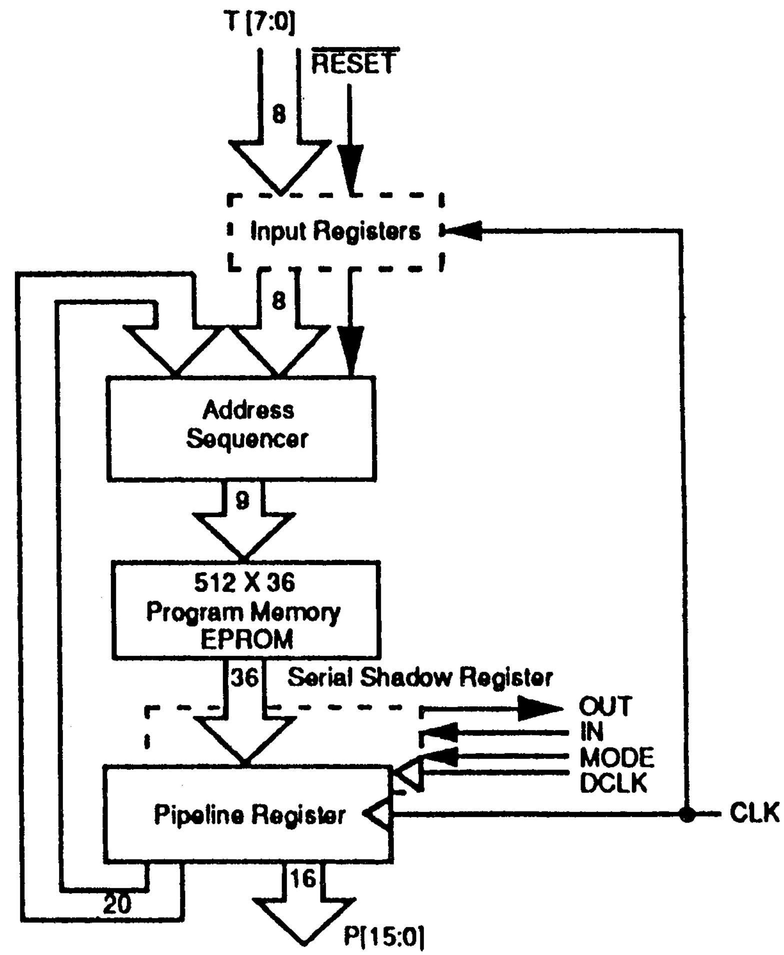
Average power consumption was measured. Subsecond power requirement was calculated on the basis of power measurements for each subsystem.

IMPROVEMENTS

PCSG -> Integrated Sequences

Entire programmable clock signal generator circuitry can be replaced by single chip field programmable controller AM29CPL154H-25-883/B produced by AMD. EPROM - 512 x 36 bits. DIP 28 I supply 115mA when active, can be switched off during exposure time.

SIMPLIFIED BLOCK DIAGRAM



DISTRIBUTED LOGIC

Distributed glue logic on CPU, PCSG, ADC, and DAC boards can be replaced by single Actel ACT 1020 field programmable gate array (883/B) in 84 pins ceramic pin grid array package. Effect ive equivalent of 2,000 gates.

CMOS - power dissipation only when active switching.

EPROM - radiation tolerant CMOS antifuse (up to 1.5 megards)

STAR TRACKER PROJECT
CURRENT STATUS

The ARC Star Tracker engineering unit was tested. It compared favorably to the standard NASA star tracker (see progress report June-August 1992). Several, but not all, flight subsystems were completed.

The project was divided into several subtasks, each one corresponding to development of a particular hardware or software subsystem. Status of each task and Star Tracker subsystem is summarized below.

Optical:

lens, fat zero illumination, athermalized sensor holder - flight hardware completed and successfully tested.

Mechanical:

interface flange, electronic compartment cover, PCB mounting, connectors mounting, Peltier coolers, thermal anchoring, LEDs, thermistors.

- design completed, flight parts manufactured and tested. Not fully assembled.

Electronics:

- design completed and tested successfully with engineering boards. Flight boards 1, 2, and 3 were successfully tested. Boards 5, 6, and 7 were tested, but require rework. Flight board #4 has not been tested.

Software:

Mini operating system:

(not modifiable after assembling) - tested operating. Problems with transmitting large amounts of data (above 500 bytes in 1 sec.) were not debugged. System was modified to allow using secondary copy of operating system which can be remotely updated.

Acquisition module:

-preliminary version is operating. Low level (hardware-interactive) procedures were completed. High level procedures (star selection, artifact rejection) are operating but not fully debugged. Acquisition module can be remotely updated.

Tracking/Commandable module:

-operating. All low level (hardware-interactive) procedures were optimized, debugged, and are finalized. High level procedures (searching, tracking, command interpreter) are operating, but were not fully debugged, and safety checks are not implemented. Tracking program can be remotely updated.

Unresolved Problems:

Lack of operational reliability of CPU boards - possible cause: bad connection or floating input.

Mini operating system failure to handle large datasets - possible cause: improper interrupt servicing.

Undeveloped and unimplemented features:

High level algorithm - acquisition, tracking and commandable modes.

Parallel hardware operation (interrupted driven) allowing simultaneous sensor exposure, data acquisition, and data processing.

Integration of the flight unit.

Instrument - level functional and environmental tests of the flight instrument.

APPENDICIES

QUARTERLY PROGRESS REPORTS

Dec - Feb 91

Mar - May 91

Jun - Aug 91

Sep - Nov 91

Dec - Feb 92

Mar - May 92

Jun 92 - Status Report and Schedule

Jun - Aug 92

Sep - Nov 92

APPLIED RESEARCH CORPORATION
PROGRESS REPORT NAS5-31169

LOW-COST STAR TRACKER

Progress Report for the Months of December 1990 - February 1991

- The characteristics of three CCD arrays with respect to their usefulness for the Star Tracker were compared:

  - TK512 (512 x 512 pixels) made by Tektronix;
  - TH7863 (288 x 384 pixels) made by Thomson-CSF; and
  - TC217 (1134 x 486 pixels) made by Texas Instruments.

We considered the following CCD parameters and determined the CCD array which we believe offers the best overall combination of these parameters:

- expected long-term availability;
- reliability;
- accuracy (star position);
- sensitivity (tracking faint stars);
- dynamic range (star magnitude);
- speed (image update rate, especially during initial star acquisition);
- flexibility of operational modes;
- no need for CCD cooling;
- low cost.

We tentatively sele ted the TC217.

- A Conceptual Design Review was held at GSFC (Tom Budney's office) on January 16. The participants were: Tom Budney (Code 745), Tom Collinson (Code 745.1), Rick Schnurr (Code 712.2), Marek Chmielowski (ARC) and Siegfried Auer (ARC). Budney verbally presented NASA's "Wish List":

  - accuracy (absolute): ~10-15 arc sec
  - update rate: ~5-20 per sec
  - power consumption: <7 watt
  - mass: <7 pounds (3.8kg)
  - star's angular velocity: up to 0.5 degree per second
  - dynamic range: -2...+6 star magnitudes,
  - a star presence signal
  - in addition to digital outputs, the following analog outputs are desired:

    (a) pitch angle (±20 arc min);
    (b) yaw angle (±20 arc min);
    (c) star magnitude (±0.25m) for the brightest star in the field of view;

Chmielowski presented ARC's conceptual design.

- A very preliminary parts list was discussed with Tim Gruner (Code 745.2).

- Tradeoffs were made between the following Star Tracker parameters:

- fast acquisition,
- high accuracy,
- high sensitivity,
- wide range of star brightnesses,
- low power consumption;
- low mass,
- high reliability,
- low cost.

The initial star acquisition will be fast (less than 1 second), while power consumption may be high and accuracy low. In the tracking mode, the update rate can be as high as 100-200 per second (increased power consumption) or as low as 1 per second (low power consumption).

- About half of the parts for the breadboard model have been ordered so far.

Problem

Texas Instruments has just (March 6) advised ARC that the fabrication of the TC217 presently yields 100 percent unacceptable devices. This problem persisted since July 1990. TI has employed a task force to resolve it. Prior to July 1990, the TC217 was sold at a high volume of about 500 per year (1989 and 1990) in the US alone.

Because of the uncertain availability of the TC217, ARC decided to go ahead with a design based on the CCD array model TC277 (732 x 290 pixels). In Phase I of this SBIR project, ARC already used a TC277 and successfully demonstrated a resolution of about 5 arc seconds and a field of view of 11 x 8 degrees. Thus, we are very confident that we can meet most if not all requirements for a star tracker using the TC277.

As soon as TI resolves its fabrication problem, we will consider modifying our design to implement the TC217 in place of the TC277. The two CCD models have very similar architecture. The TC277 is being sold at a very high volume of about 3000 per year in the US alone, and there appears to be no problem with fabrication and supply over the foreseeable future.

An alternative CCD array would be the TC213 or TC215 (1024 x 1024 pixels). Because ofits much higher cost ($5,500 versus $690 for the TC217 and $585 for the TC277) and very low production volume, we consider it as a backup to the TC277.

Planned Activities for the Months of March, April, May 1991

- To assemble the circuits to drive the CCD array, to amplify the video signal, to digitize the video signal, to store the digitized signals, and to test and operate this circuitry with a TC277. As drive circuits, we plan to initially use commercial non-flight IC's from TI and to later replace them with parts which are available in high-rel. As the video amplifier, we plan to initially use a circuit design developed by the Kitt Peak National Observatory and to modify it for low power consumption and high reliability. For all other circuits, we plan to use the commercial equivalent of the high-rel parts that we intend to use in the protoflight model. A preliminary parts list is enclosed.

- The electronic readout will be first tested with IBM-PC based digital I/O card. This will allow testing of firmware and low level blocks of software before implementing into dedicated microprocessor system and into PROM.
- To set up a microprocessor development system for the 80C86.
- To measure the focusing characteristics of several lenses.

Preliminary Parts List, March 1991 ARC Star Tracker

Part	Function	High-rel.	Rad-hard
80C86	CPU	ISTP	X
5012	ADC, 10µs	sim. to 5016 (ISTP)	X
or 7672	ADC, 5µs	883	
CLC505	high speed op amp	S level	
OPA404	quad FET op amp	883	
LM139	voltage comp.	ISTP	
A-250	charge sens. preamp	ISTP	X
6617	2K x 8 CMOS PROM	ISTP	X
6564	8K x 8 RAM	ISTP	X
82C54	interval timer	ISTP	X
82C55	interface	883	X
82C82	octal latch	883	X
82C85	clock controller	ISTP	X
HCS series	high speed CMOS logic	ISTP	X

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: March - May 1991

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 Corporate Drive
Suite 1120
Landover, MD 20785
(301) 459-8442

We received one CCD array, Texas Instruments' model TC 217-40. Meanwhile, TI has resolved its manufacturing problems and is shipping two more CCD arrays, model TC 217-30, the highest grade available.

The electronic circuits to drive the CCD array, to amplify and to digitize the video signal were assembled and tested with the TC 217 CCD array.

As drive circuits, we used commercial non-flight-qualified ICs from TI (TMS3473B and SN28846).

In parallel, we are developing CCD drive circuits which meet the following stringent requirements:

1. Quiescent power must be minimal (≈ 1 mW per driver); this is a long way from the typical quiescent power requirement of the order of 0.3 - 0.5 W for a commercially available high-rel CCD driver.

2. Only parts are to be used which have high-rel, rad-hard counterparts.

3. A driver must be capable of driving a capacitive load of 13,000 pF with signals having a 12 V amplitude and a fall time of only 90 ns, or driving a capacitive load of 1100 pF with a rise/fall time of only 15 ns.

4. Adjustable, intermediate voltage levels must be provided for 2 of the 9 drive signals. Commercially available high-rel CCD drivers have no such provision.

5. The HIGH and LOW voltage levels for 6 of the 9 drive signals must be individually adjustable for optimum CCD performance, yet the driver inputs must be HCMOS levels (fixed at 0 V and + 6 V). Commercially available high-rel drivers have no such provision.

Table 1 lists the relevant requirements for each signal. Preliminary tests indicate that most or all of the above requirements will be met by our circuits.

As the video amplifier, a modified version of a circuit developed by the Kitt Peak National Observatory was built and tested.

We digitize the video signal using a commercial AD7672.

All electronic circuits were designed to be directly controlled by TTL signals generated by an I/O card in an IBM-PC compatible computer. This allows testing of firmware and low and high level blocks of software before implementing into dedicated microprocessor system and into PROM.

To date, several parts of software were developed and tested:

1. The acquisition module - part of software directly interacting with hardware. This module was written in assembly language and allows direct control of CCD drivers, video amplifier and A/D converter. The major function of this package is to digitize a selected part of the CCD image and store the selected "window" in the computer memory.

   This package also allows for continuous removal of charge generated in CCD to prevent saturation. This function is executed in parallel with other computer tasks.

2. The image analysis module - part of software used for calculating star positions, selecting "windows" to be digitized and adjusting operational parameters.

   This part of software is currently under development. The module calculating a single star position is now tested. The modules for selecting operational parameters and tracking multiple stars are not fully operational yet.

The above described hardware and software were tested using a He-Ne laser and a black-body radiator simulating starlight.

Preliminary results suggest that the entire system noise will allow a resolution better than 1/10 of the CCD pixel size.

TABLE 1 CCD INPUT SIGNALS FOR TC 217

Clock Gate	Capacitance	Rise Time	Fall Time	Voltage Swing	Inter-mediate Level	Voltage Levels Requiring Individual Adjustment
Image Area Gate	13,000 pF	150ns	90ns	-10V...+2V	-2V	-2V
Antiblooming Gate	1,100 pF	150ns	90ns	-7V..+4.5V	-1.2V	-1.2V
Transfer Multiplex Gate	150 pF	15ns	15ns	-10V...+2V	-	-10V, +2V
Storage Area Gates 1 & 2	11,000 pF	150ns	90ns	-10V...+2V	-	-10V, +2V
Transfer Gate	200 pF	150ns	90ns	-10V...+2V	-	-
Serial Register Gates 1, 2 & 3	180 pF	15ns	15ns	-10V...+2V	-	-10V, +2V

PLANNED ACTIVITIES FOR THE MONTHS OF JUNE, JULY & AUGUST 1991

1. Develop software for localizing and tracking of single simulated star, and for automatic adjustment of operational parameters of hardware.

2. Test tracking system linearity and noise using simulated star and rotational camera mounting.

3. Select acquisition parameters to optimize non-linearity and random noise errors. Develop linearization procedures if necessary.

4. Expand software to allow localizing and tracking multiple stars.

5. Test time of execution at different parts of software followed by software optimization with respect to the execution time.

6. Design PCB for CCD drivers, assemble and test drivers with CCD array.

7. Design CCD and lens holders. Holders will allow for compensation of difference in temperature expansion coefficients between CCD array and lenses.

Preliminary Parts List, June 1991

Part	Function	High-Rel	Established Rad-Hardness
CF 50 N	Camera Lens, 50 mm/f2		10 Mrad
FRL 130R	Power MOSFET (≈IRFF 133)	SAMPEX	yes
FRL 9130R	Power MOSFET (≈IRFF9133)	SAMPEX	yes
80C86	CPU	ISTP	yes
82C54	Interval Timer	ISTP	yes
82C55	Interface	883	yes
82C82	Octal Latch	883	yes
82C85	Clock Controller	ISTP	yes
5012	12 bit ADC (≈5016, ISTP)	(ISTP)	yes
7672	12 bit ADC (alt. to 5012)	883	
HA-5147	Low Noise High Speed Op Amp	883	
OPA-602	High Speed FET Op Amp	883	yes
LM 139	Voltage Comparator	ISTP	
A-250	Charge Sens. Ampl.	ISTP	yes
1840	16 Ch. Multiplexer	ISTP	yes
LM 124	Op Amp	PPL 19	
MAX 543	12 bit DAC		
REF 02	Volt.Ref.& Temp.Sensor	ISTP	
OW 62256	32kx8 SRAM	SAMPEX	yes
AC 00	NAND Gate	SAMPEX	
AC 02	NOR Gate	SAMPEX	
AC 14	Schmitt Trigger	SAMPEX	
4053B	CMOS Analog Switch	883	
4066B	CMOS Analog Switch	PPL 19	
6617	2k x 8 PROM	ISTP	yes
HCS series	High-Speed CMOS Logic	ISTP	
1N5711	Schottky Diode	PPL 19	
1N4148	Diode	PPL 19	
2N2222	npn Transistor	PPL 19	
2N2907	pnp Transistor	PPL 19	

/ 2-ue

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: June - August 1991

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 Corporate Drive
Suite 1120
Landover, MD 20785
(301) 459-8442

We developed a system that permits testing the linearity and positioning error of the star tracker. The test system consists of a TI model TC 217 CCD array and a Nikon 50 mm/1.2 lens, mounted on a motorized three axis rotator (Oriel model 18368) with computer-controlled positioning.

The following software modules have been developed to record the position of a simulated star while rotating the camera.

1. Star finding module: That part of the software which initially finds the star in the field of view, selects the tracking window and the initial exposure parameters. This module was tested using various image compression techniques in order to speed up the procedure.

2. Star tracking module: That part of the software that continuously tracks the star, repositions the tracking window and adjusts the exposure parameters.

3. Star position analysis module: That part of the software which estimates the precise star position (to a fraction of a pixel), the position error, the star magnitude, etc. This procedure utilizes data from both non-interlace and interlace (frame shifted by ½ pixel) modes to permit comparing the star tracker accuracy achievable in various modes of CCD operation.

4. Scanning module: That part of the software that controls the angular pointing of the camera. This module allows scanning in 2 axes with an accuracy of better than 0.05 pixel (with an f=50 mm lens) while the star position is being recorded. A variable-size step technique was used to reduce the amount of data to be recorded while preserving small and large scale accuracy. The backlash compensation characteristics of the rotational stages were taken into account while tracking a simulated star and recording the data.

5. Background compensation module: That part of the software which controls the "fat zero" (due to background illumination) of the CCD array and eliminates the influence of the background on the estimated star position. This module is still under development.

The test results (see figure 1) indicate that the error and the reproducibility of the estimated position are about 0.05 pixel in the horizontal axis of the CCD array (corresponding to a resolution of 1.5 arc sec within a field of view of 10 degrees). The cumulative (when scanning in both horizontal and vertical direction) non-compensated nonlinearity is about 0.2 pixel.

In the vertical axis of the CCD array, the interlace technique (frame electrically shifted by ½ pixel) is used to increase the resolution. However, when using this technique, a strong non-linearity in dark current and sensitivity in the vertical direction is introduced. We are now developing software for compensating these undesirable effects.

Preliminary CCD sensitivity measurements, using both wide spectral continuum radiation from a blackbody and discrete color from a He-Ne laser, have been made and the quantum detection efficiency was estimated using both thermopile and silicon detectors. Equipment for a full characterization of the star tracker's spectral sensitivity is being set up.

We developed and partially tested a CCD drive circuit that will probably meet all of the stringent requirements listed in the last report (i.e., minimal quiescent power; use of high-rel and rad-hard components; very fast switching; intermediate voltage level; and input compatible with HCMOS).

A PCB layout for the drive circuit has been completed. We expect the circuit performance to depend on the PCB layout; therefore, a full and realistic test can only be made with the assembled PCB. We plan this as the next step after PCB manufacturing.

We are working on the mechanical and thermal design of the housing and support of the lens and the CCD array and the flange by which the star tracker is mounted to the spacecraft. The CCD mounting fixture is designed to hold the CCD array such that the center of its image area stays on the optical axis within about ± 1 μm (corresponding to ± 4 arc sec) over the entire operational temperature range (-10 to +50 degrees Celsius).

Planned Activities for September, October, and November 1991

1. Test the tracking system's linearity in the vertical direction of the CCD and optimize the exposure parameters to minimize the influence of the pixel centroid shift on the linearity.

2. Develop linearization algorithm.

3. Test position error within different portions of the CCD image area and for different CCD chips.

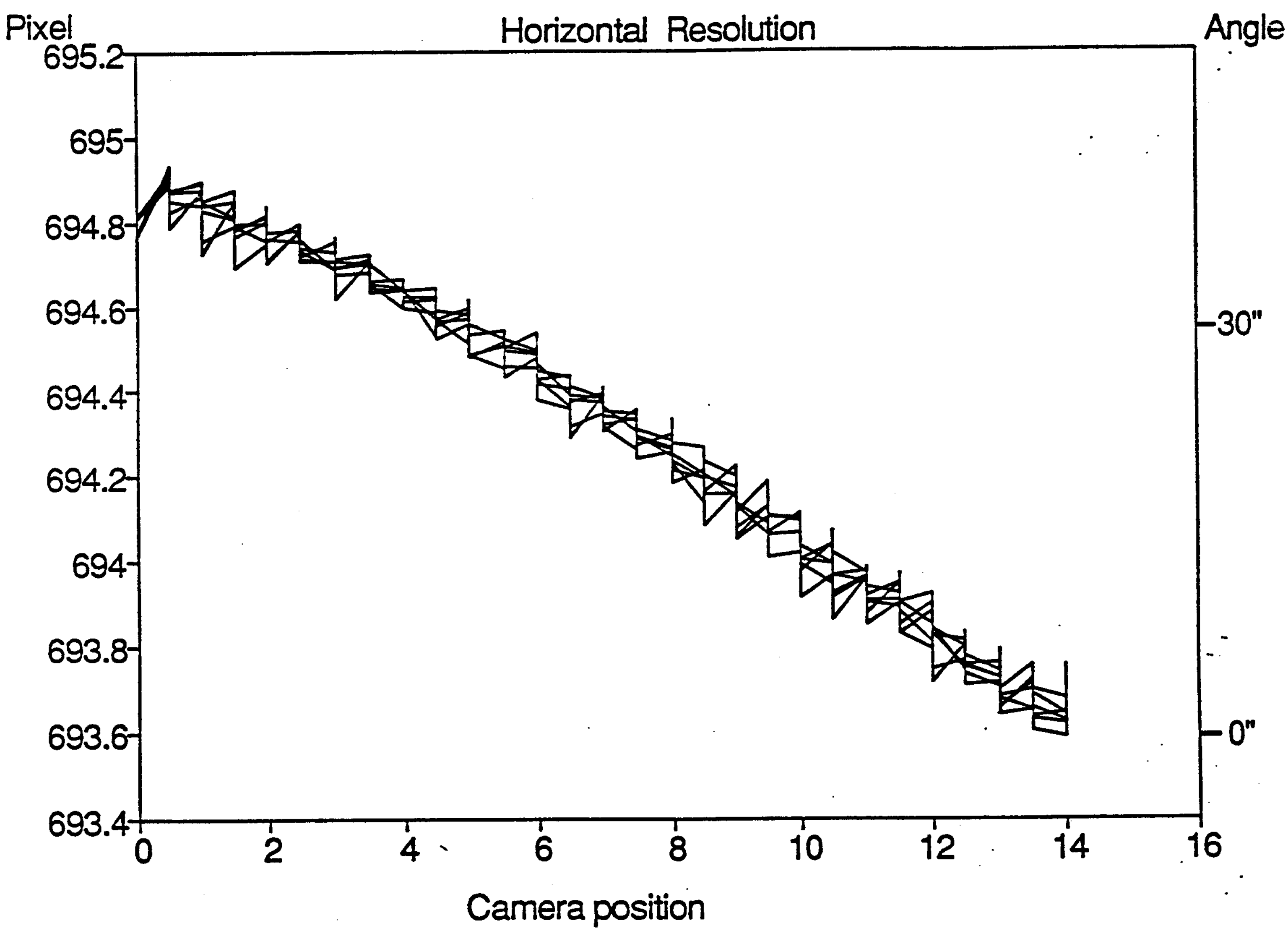
4. Characterize the CCD sensitivity and the position error as a function of wavelength.

5. Expand software to allow finding and tracking up to five stars.

6. Test time of execution at different parts of software and optimize software with respect to the execution time.

7. Complete the design of CCD and lens holders and of the mounting flange. The holders will allow for compensation of differential thermal expansion coefficients.

8. Prepare and present Preliminary Design Review.


Horizontal Resolution
Pixel
Angle
695.2
695
694.8
694.6
694.4
694.2
694
693.8
693.6
693.4
30"
0"
0
2
4
6
8
10
12
14
16
Camera position

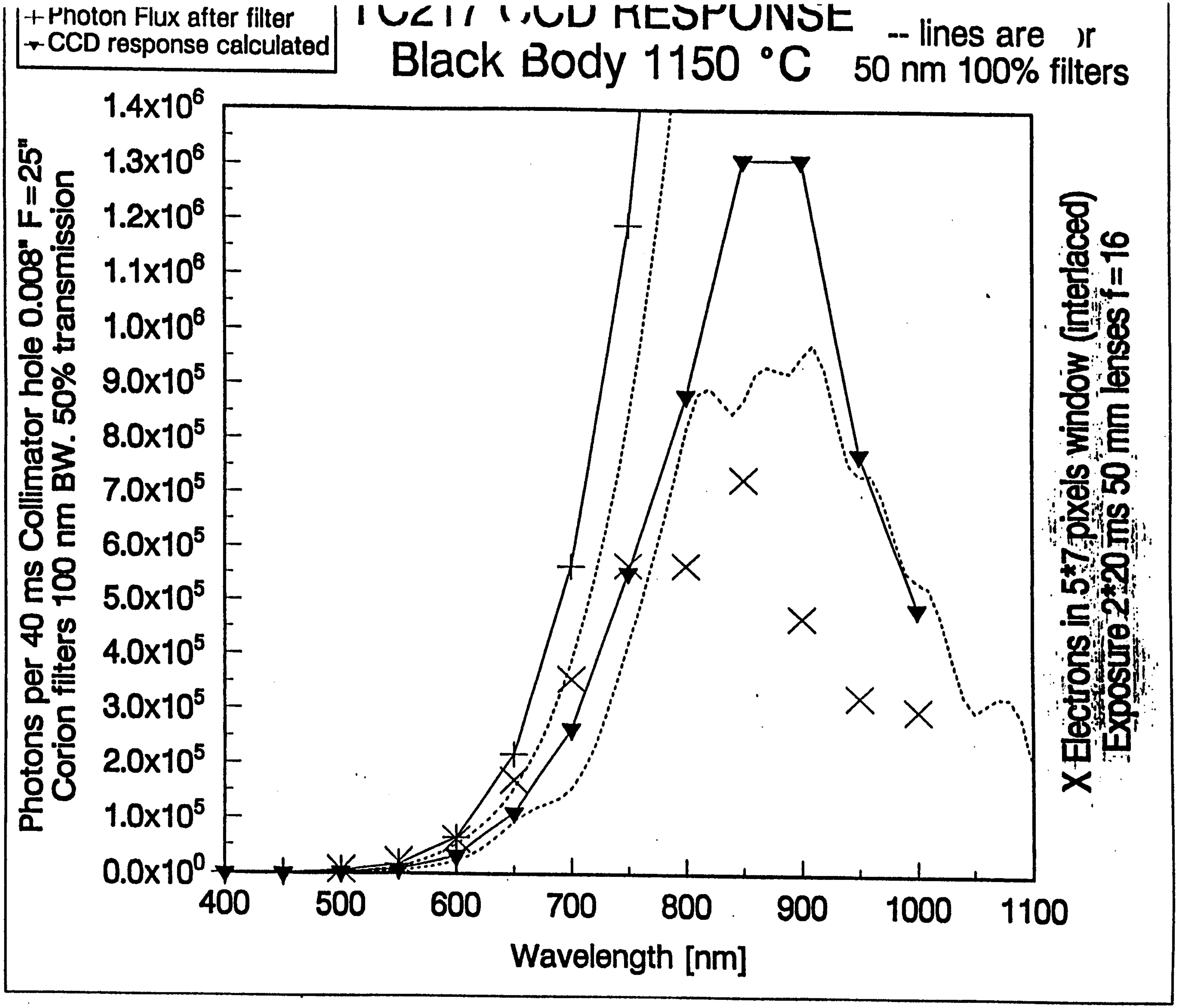
PROGRESS REPORT

LOW-COST STAR TRACKER

Period: September - November 1991

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 Corporate Drive
Suite 1120
Landover, MD 20785
(301) 459-8442



+ Photon Flux after filter
CCD response calculated
TC217 CCD RESPONSE
Black Body 1150 °C
-- lines are for
50 nm 100% filters
Photons per 40 ms Collimator hole 0.008" F=25"
Corion filters 100 nm BW. 50% transmission
1.4x10^6
1.3x10^6
1.2x10^6
1.1x10^6
1.0x10^6
9.0x10^5
8.0x10^5
7.0x10^5
6.0x10^5
5.0x10^5
4.0x10^5
3.0x10^5
2.0x10^5
1.0x10^5
0.0x10^0
400
500
600
700
800
900
1000
1100
Wavelength [nm]
X Electrons in 5*7 pixels window (interlaced)
Exposure 2*20 ms 50 mm lenses f=16

Progress:

The spectral sensitivity of the CCD (TC217) and the Nikon 50 mm/1.2 lens were characterized using a temperature stabilized blackbody source and narrow-band visible infrared interference filters. In the wavelength range 400 - 750 nm reasonable (better than 30%) agreement between calculated and measured data was found (see Figure 1). In the range 750 - 1000 nm the measured response is consistently 50% lower than expected.

To find out the source of this discrepancy the response of the CCD without lens was compared to the response of a thermopile detector. The measured response of both thermopile detectors and CCD without optics have been found to agree within 20% of the expected value. Thus, the discrepancy in long wavelength ranges was attributed to a lower lens transmission in this wavelength range.

The linearity of centroid algorithms was tested as a function of tracking window size, lens f number, CCD position, and wavelength. A window size as large as 9x15 pixels was found to be critical for obtaining linearity errors smaller than +/- 0.1 pixel (equivalent to +/-3 arcseconds). The remaining parameters were found to have a negligible effect on linearity errors and position noise.

Data taken with a 9x15 window exhibit position modulation with one pixel period. The modulation amplitude is about 0.08 pixel (about 2.5") and stable in time. A Fourier correction algorithm was applied to eliminate this modulation. Only a one-dimensional correction algorithm was applied up to now. The evaluation of the algorithm's final accuracy as well as the evaluation of a two-dimensional correction algorithm were postponed until the CCD mounting flange will be ready and thus the mechanical stability will be assured.

The TI CCD exhibits relatively large charge trappings at room temperature. To allow detection of faint stars it was necessary to fill the charge traps at the beginning of the exposure times. A 'FAT ZERO' circuit based on LEDs was developed and tested.

CCD, Integrating Amplifier, Far Zero circuit, and A/D converter have been successfully tested in the temperature range from -10 to +40° C. Above +40° C high charge spreading was observed.

Simplified versions of the integrating amplifier have been tested. However, the charge injection due to unbalanced switches was found to be unacceptably large.

The relative processor time necessary for execution of different parts of software was measured and it was found that 98% of the processor time was spent on repetitive generating of various waveform patterns. The Programmable Clock Signal Generator (PCSG) is intended to generate some of these waveform patterns thus leaving the CPU free to perform calculations of the star position. There is a trade-off between the complexity of the PCSG and the amount of tasks which can be transferred from the CPU to the PCSG. Thus, it was necessary to perform a detailed analysis of the software execution times and of the complexity of the design before preparing design specifications for the PCSG.

The specifications of the digital circuits of the Star Tracker (CPU and PCSG) have been prepared and submitted to electrical engineers.
Following a recommendation by Peter Corey of Nikon, we dropped our plan to use a Nikon lens for the Star Tracker. Instead, we issued a subcontract to Tucson Optical Research Corp. to custom-design a lens and a lens mount and to manufacture a prototype lens. This lens will use only radiation-resistant glasses. So far, the optical design has been completed, see the enclosed report.

<u>Planned Activity for December 1991 through February 1992.</u>

1. Design and test of auxiliary electronics -- backup and housekeeping A/D converter, D/A converters used for on-line adjustment.

2. Test of final version of Analog Electronics, CCD drivers, auxiliary electronics, over the -10° to +50° C temperature range.

3. Prepare final design of electronics -- analog, auxiliary, and digital.

4. Prepare final mechanical design.

5. Transfer software from IBM PC computer to dedicated CPU board.
6. Prepare test software which will simulate spacecraft Host Computer.

7. Design lens mount and fabricate prototype lens.

8. Prepare and present preliminary design review.

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: December 1991 - February 1992

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 Corporate Drive
Suite 1120
Landover, MD 20785

(301)459-8442/FAX (301)731-0765

PROGRESS:

The CCD driver circuit design was finalized, built in breadboard and PC board versions, and successfully tested in the temperature range from -20 to +50°C. As opposed to previous tests with TI commercial drivers, our drivers operated properly, even in the temperature range from +40°C to +50°C.

However, at elevated temperatures, the increased CCD dark current limits the performance of the CCD sensor. So, we tested different thermoelectric coolers and intend to use them to keep the CCD temperature below or at room temperature.

Auxiliary electronics (temperature monitoring, housekeeping, D/A converters, analog outputs etc.) were designed, built in breadboard version, and tested. The circuit can monitor in flight: temperatures at 4 different locations, the CCD output and the amplified signal at 3 points, 16 different supply and bias voltages. To save power, a switch for turning off most idling circuits during exposure time was designed, built in breadboard version and successfully tested at frame rates up to 10Hz.

A power supply was designed with Interpoint MHF DC/DC converters (883 classification pending). Due to relatively high cost, converters have not been purchased and tested. We intend to test them after NASA acceptance during preliminary design review. Remaining elements of power supply have been tested at breadboard level as well as an alternate power supply based on step-down converters.

The driver and amplifier board was designed, the PCB layout was prepared, the board was manufactured, assembled and successfully tested at room temperature.

The analog/digital interface board was designed, the layout is currently being prepared. CPU and PCSG electronics have been designed, and breadboard versions are now under debugging and testing.

The mechanical design of main elements was finalized.

An athermalized lens mounting was designed. A temperature and thermal stress compensated CCD mounting was designed. The cover design will be finalized after completing the entire electronic design.

A prototype version of the lens (based partially on non radiation hard glasses) is currently produced.

Methods of software transfer from IBM-PC computer to embedded system was selected and appropriate software (locator TDREM, Run Time Lib) purchased.

Responding to a verbal request from Mr. T. Collinson, we redesigned the analog interface .

Planned Activities for March 1992 through May 1992.

1. Preliminary Design Review - implementing requested modifications.

2. Programming and testing digital electronics.

3. Design board layout for digital electronics, assembly and testing.

4. Develop multi-star tracking software.

5. Assemble entire electronics.

6. Assemble star tracker.

7. Prepare operational version of software to be transferred to embedded system.

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: March 1992 - May 1992

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 Corporate Drive
Suite 1120
Landover, MD 20785

(301)459-8442/FAX (301)731-0765

PROGRESS:

A prototype of the custom made lenses was produced and tested. Problems concerning proper location of focal plane and proper size of the reflector for "fat zero" LED were corrected. Acquisition/tracking software was modified to match characteristics of new optical system. The final version of mounting flange was designed. The cover design was finalized.

The analog/digital interface board was assembled in PC board version and successfully integrated with amplifiers/drivers board.

PCSG generator was assembled in breadboard version, tested and correction were implemented. Software generated wave forms were transferred to PCSG and tested. Acquisition software were modified to use both PCSG and software generated waveforms. Electronic circuit consisting with driver/amplifier, analog/digital interface, PCSG boards is currently tested with IBM-PC computer. Basic acquisition/tracking mode of operation was already tested. Fast search is under debugging.

Multi star acquisition/tracking software were developed and tested in one exposure time version. Multi star acquisition with multiple exposure times (to track stars of substantially different magnitudes) is developed.

Operational version of software to be transferred to embedded system was prepared. Hardware control signals, software 1/0 space and interrupt services were converted to be dual system (IBM-PC and embedded processor) compatible. This software will be expanded by adding modules which are currently developed.

Final PC board design was prepared for amplifier/driver, analog/digital interface and PCSG boards. Prototype of power supply housekeeping board is now assembled.

CPU board was partially assembled in breadboard version and debugged. Programming kernel for software transfer to embedded system was developed, stored in ROM and debugged.

Preliminary Design Review was prepared and submitted to NASA.

PLANNED ACTIVITY FOR JUNE 1992 - AUGUST 1992:

1. Finalizing operational version of software.
2. Finalizing design of CPU board ~~power~~.
3. Testing power supply/housekeeping board.
4. Prepare final documentation and layout of electronic - flight boards production.
5. Transferring software to embedded system - debugging.
6. Developing communication software and host computer simulator.

Low Cost CCD Star Tracker
Contract NAS5-31169
ARC 1150

Status Report and Schedule

Status Report

Up to date (17th July 1992) the following tasks have been accomplished (marked if only partially).

- Design
  - Conceptual
  - Electronics
  - Algorithms
  - Optical
  - Mechanical (modification is expected)
  - Calibration procedures
- Prototyping and Testing
  - CCD functional test
  - CCD operational parameters limitation
  - Optics
  - Electronics (except CPU board)
  - Software - hardware interacting routines (except specific for CPU board)
  - Algorithms in IBM version
  - Calibration procedures (tested only with preliminary optics)

For time frame of accomplished tasked see Figure Schedule - full triangle marks.

Configuration and Performance

Star tracker is currently operated in prototype version consisting of all analog and mixed analog and digital boards in printed circuit version (PCB), with one digital board in prototype version, and with IBM computer controlling entire operation (in place of dedicated CPU board).

Entire electronics were tested in various operational condition including over temperature, under and over voltage etc. The final (expected to be) version of PCB's are in layout preparation stage (except CPU board).

Functional operation of the entire electronics, software routines interacting with hardware, and basic algorithm were tested. Star tracker is capable to detect, locate, track and report position of multiple stars. Exposure time and CCD parameters are automatically adjusted accordingly to external (illumination) or internal (temperature) conditions.

Hardware interacting routines as well as calculation modules are in final version (except CPU board specific routines) while software governing algorithm execution is basic form i.e., all star tracker functions like: searching, locating, and tracking of multiple stars are successfully executed but improvements should be done to increase performance.

To Be Done

The key step is to test the proper operation of CPU board - the last to be tested. The delivery of wire wrap prototype of CPU board is expected in the next few weeks. After testing CPU board the final version of PCBs for entire electronic can be ordered as well as software routines interacting with hardware can be fully converted to final form. Operational CPU board is also necessary for improving software performance as timing of CPU board is substantially different than that of IBM. The following list of task to be done as well as schedule is prepared on the assumption that CPU board prototype will require only **minor modifications** and that ROM based software (delivered with CPU board) will **assist transferring software** from IBM to embedded processor **as expected.**

Tasks to be done are organized to ensure hardware production in final version and possibly in flight quality version by the end of November 1992. To that date the software will be developed in version ensuring star tracker operation with acceptable performance, however, the further software development will be necessary to increase performance to the design specified level.

To Be Done

- CPU board test
- ROM based software tests
- Software transfer to embedded processor
- Preparing final documentation of all PCBs
- Board production and assembly in final version
- Test and calibration of final version of entire star tracker hardware
- Software development

o Assembly of flight version of star tracker

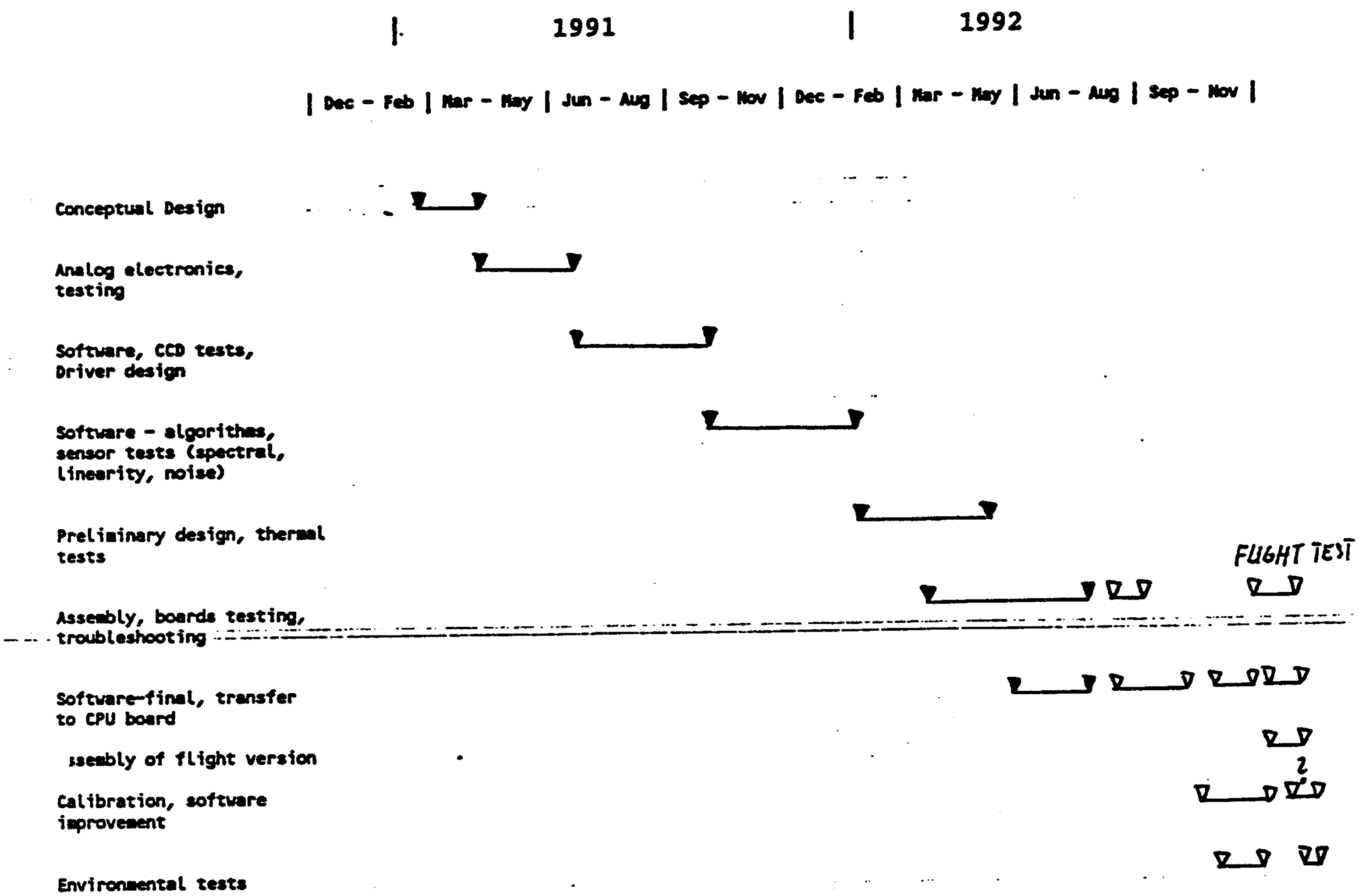
o Environmental tests

o Software improvement

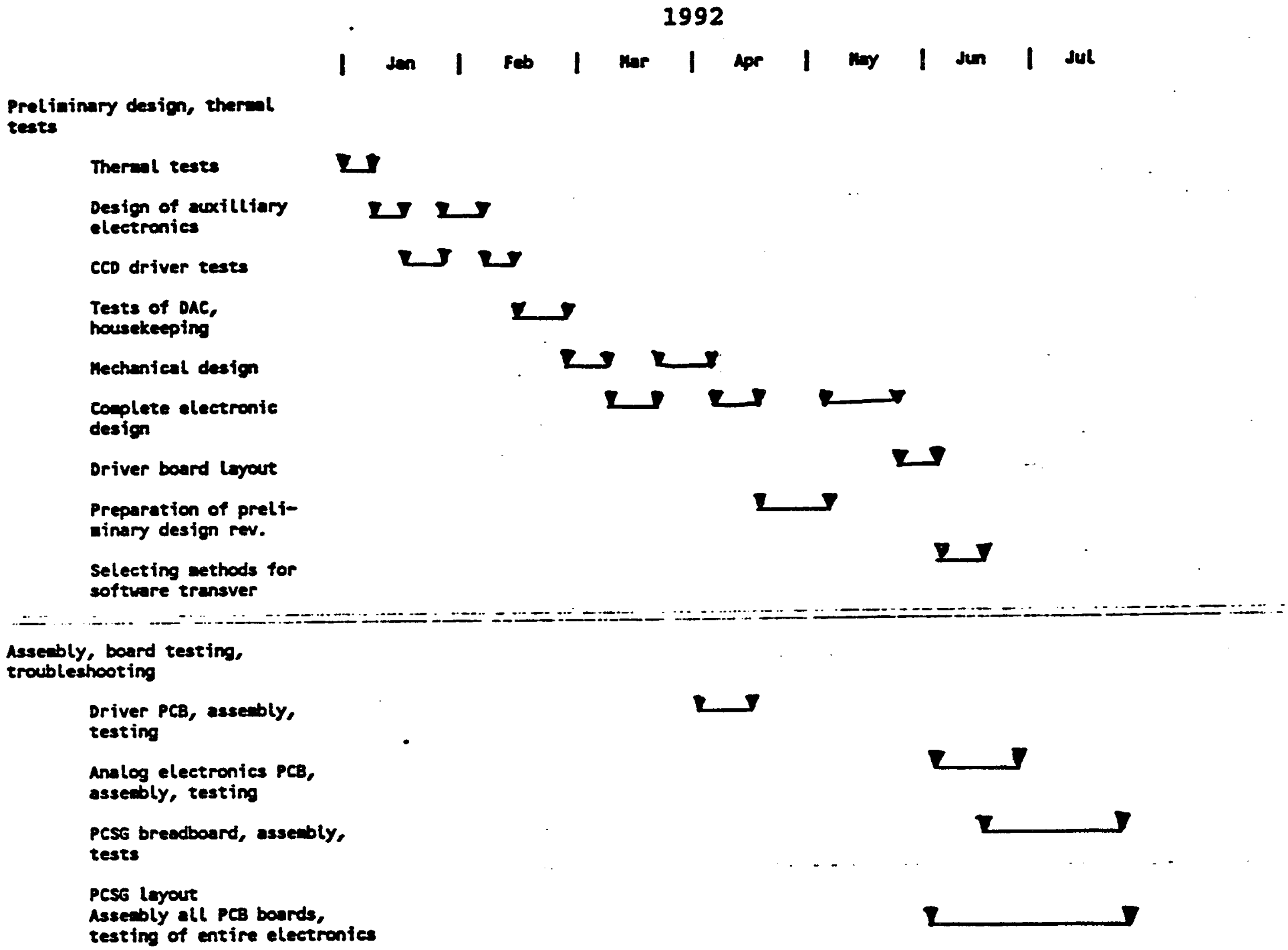
Open circles marks task which may not be accomplished on time.

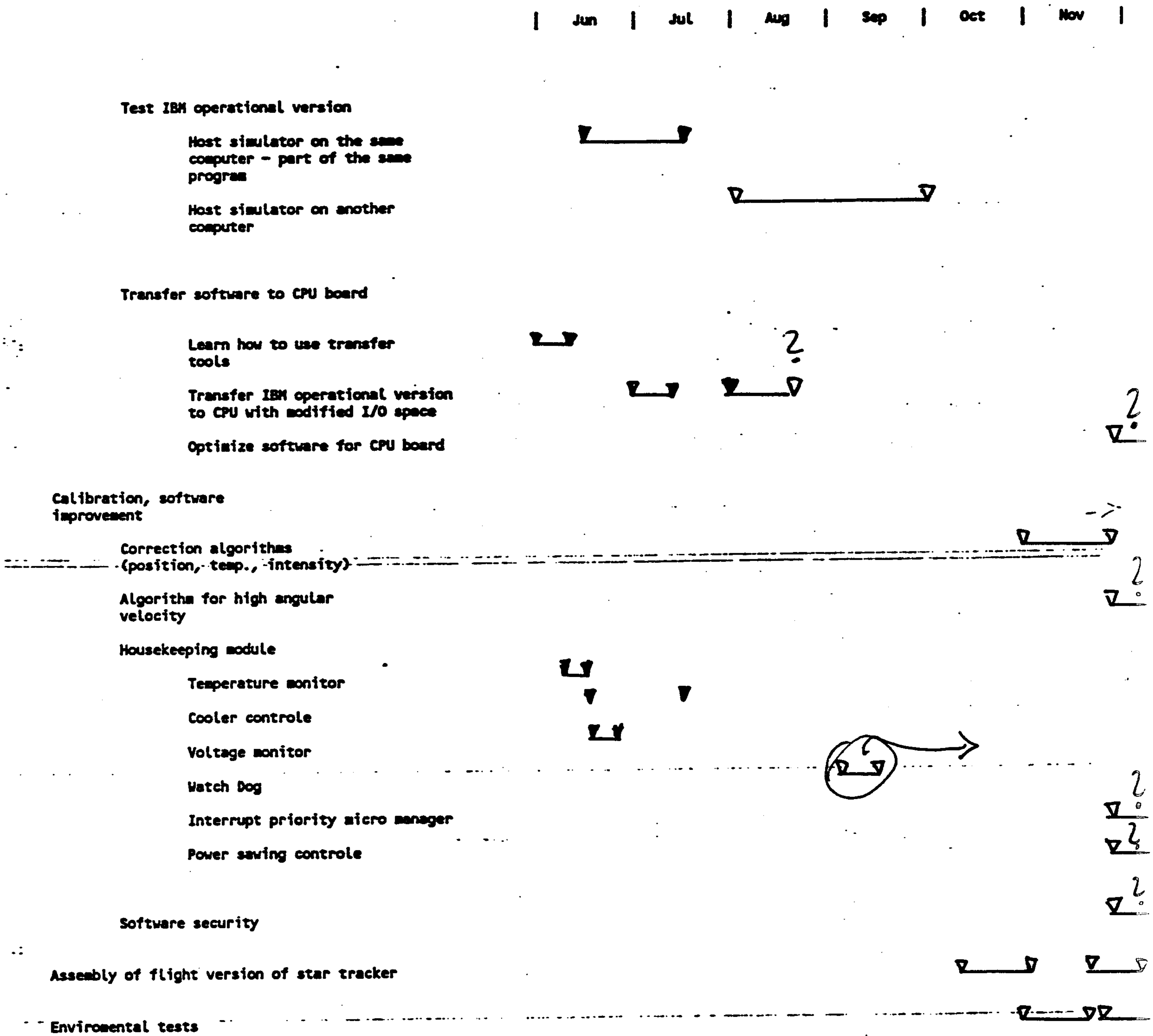
For time frame of tasks to be done see Figure Schedule - open triangle marks.

SCHEDULE - OVERVIEW

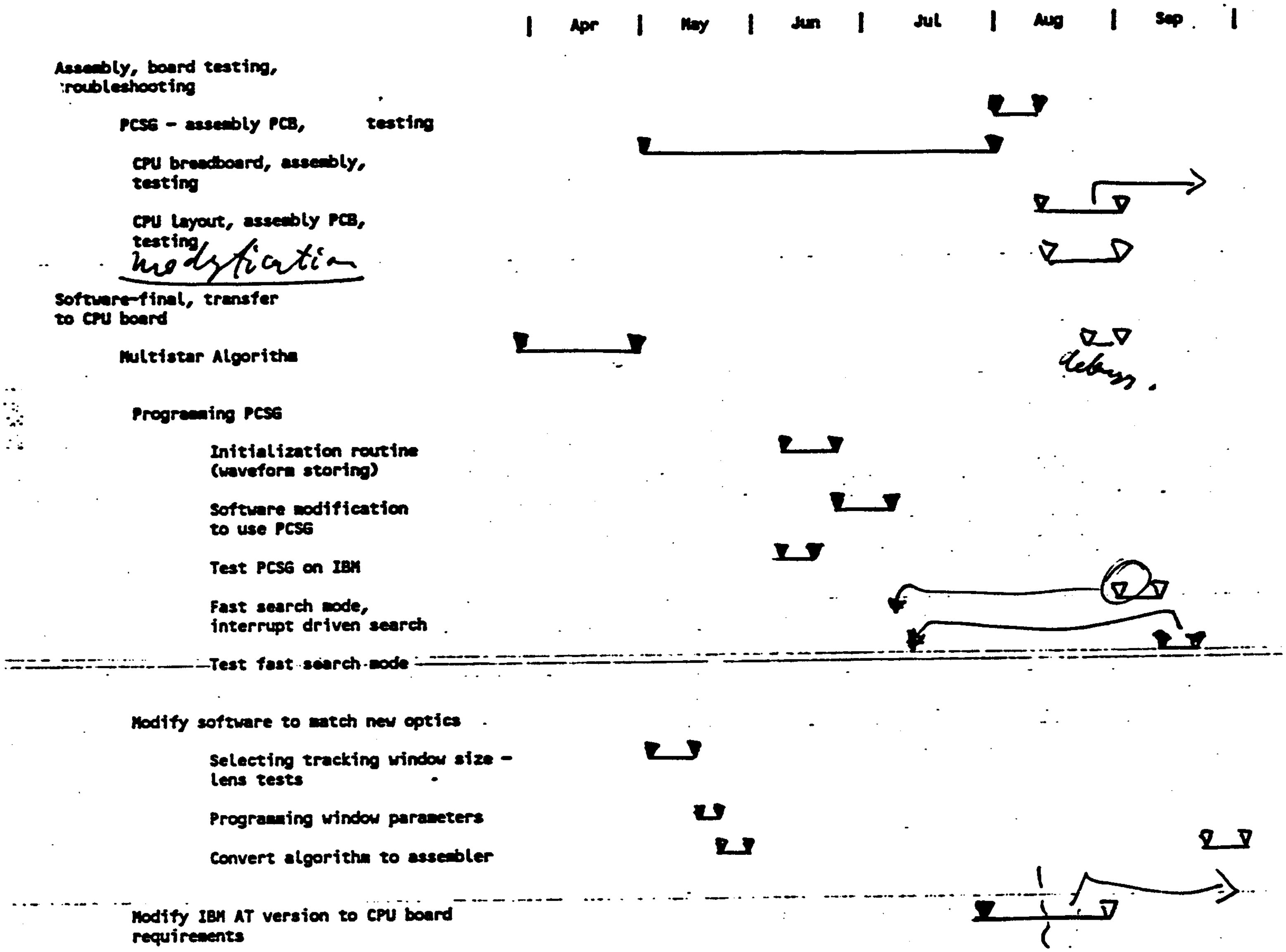


SCHEDULE - DETAILS





Jun
Jul
Aug
Sep
Oct
Nov
Test IBM operational version
Host simulator on the same computer - part of the same program
Host simulator on another computer
Transfer software to CPU board
Learn how to use transfer tools
Transfer IBM operational version to CPU with modified I/O space
Optimize software for CPU board
Calibration, software improvement
Correction algorithms (position, temp., intensity)
Algorithm for high angular velocity
Housekeeping module
Temperature monitor
Cooler controle
Voltage monitor
Watch Dog
Interrupt priority micro manager
Power saving controle
Software security
Assembly of flight version of star tracker
Enviromental tests


Apr
May
Jun
Jul
Aug
Sep
Assembly, board testing, troubleshooting
PCSG - assembly PCB, testing
CPU breadboard, assembly, testing
CPU layout, assembly PCB, testing
modification
Software-final, transfer to CPU board
Multistar Algorithm
debug
Programming PCSG
Initialization routine (waveform storing)
Software modification to use PCSG
Test PCSG on IBM
Fast search mode, interrupt driven search
Test fast search mode
Modify software to match new optics
Selecting tracking window size - lens tests
Programming window parameters
Convert algorithm to assembler
Modify IBM AT version to CPU board requirements

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: June - August 1992

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE
SUITE 1120
LANDOVER, MD 20785
(301)459-8442
(301)731-0765 FAX

PROGRESS:

An engineering model of the star tracker was assembled and tested. It consists of: custom designed lens, mounting flange, and electronics (driver/amplifier, analog/digital interface, and power supply boards all in PCB version, as well as programmable clock signal generator board in wire wrap version). The software is performing the following basic star tracker functions: acquisition, tracking, and mapping. This was tested on an IBM-PC, interfaced to the engineering model.

Multi-star acquisition and tracking operations as well as its self-testing capability were demonstrated to NASA. The star tracker performance was compared to NASA star tracker. The ARC star tracker was found operating with full performance with simulated stars of -1 to +4 star magnitude. The noise equivalent angle (for a stationary star) was found to be better than 3 arc seconds in the above mentioned star magnitude range.

All seven boards of the star tracker electronics were designed, the layout was prepared for four PCBs and these boards are now being manufactured. The CPU unit was designed and a wire-wrap version was electrically tested. The layout for three CPU boards is now in preparation.

The software transfer to the embedded system is in progress: ROM software was prepared and tested, hardware related routines were transferred and tested. The main star tracker program is being revised.

PLANNED ACTIVITY FOR SEPTEMBER 1992 - NOVEMBER 1992

1. Prepare final documentation.
2. Modify software to improve tracking speed.
3. Develop communication software and external command interpreter.
4. Transfer entire software to embedded system.
5. Assemble and test flight version of star tracker.
6. Perform environmental tests and calibration.

PROGRESS REPORT

LOW-COST STAR TRACKER

Period: September to November 1992

CONTRACT NAS5-31169

APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE
SUITE 1120
LANDOVER, MD 20785
(301)459-8442
(301)731-0765 FAX

PROGRESS

The layouts for three CPU boards were prepared and the boards were manufactured. All seven flight-version PCBs of the star tracker were assembled. Each board was electrically tested; layout and assembly errors were corrected. Boards #1, #2, and #3 (CCD drivers and amplifiers, A to D conversion and logic, power supplies and self testing) were interconnected and successfully tested. Boards #4, #5, #6, and #7 (sequencer, CPU, memory, serial interface) were interconnected and tested. Several design, layout, and assembly errors were found and corrected. Boards #5, #6, and #7 functioned with serious reliability problems, which have not yet been fully debugged. Functional tests on board #4 were not performed because such tests are not feasible without Boards 5 to 7 operating properly. The star tracker interface flange was redesigned to improve thermal contact with the Peltier coolers. The redesigned flange was fabricated and tested to demonstrate mechanical compatibility with other flight hardware. The serial interface was placed on a separate PCB, as requested by NASA. The electronics compartment housing was redesigned and manufactured, and the mechanical compatibility was demonstrated. The mini operating system (ROM) was modified to accommodate hardware changes, tested, and several bugs found and corrected (memory tests, software overwrite protection, command set). Errors causing unrecoverable problems when transmitting large amounts of data have been localized but not yet corrected. The system was modified to allow using a secondary copy of the operating system which can be remotely updated. The main star tracker program was simplified, and diagnostic code was partially removed. The entire program was remotely loaded into EEPROM (using the serial link) and successfully tested on the engineering model. The program was also tested on flight hardware to the extent possible. Summary documentation was prepared and the project status was presented to NASA representatives.

SOFTWARE LISTING

Required Software Packages

Communication Kernel and bootstrap code	
Turbo Assembler listing	CCD.Lst
Loading program	
Turbo C Listing	PLoad.C
Star Tracker Program	
Star Tracker	St5.Pas
Units	
Start	STStart.Pas
Definition	STDef01.Pas
Acquisition Setup	STAQS01.Pas
Acquisition	STAQ01.Pas
Main Control	StMain01.Pas
CCD Data Processing	StPpic01.Pas
Centroid Calculating	StCpic01.Pas
Diagnostic Display	StDpic04.Pas
Sequenser Programming	StPCS601.Pas
Self Diagnostics	StTest.Pas
Initial Search and Settings	STSrch01.Pas
Housekeeping Control	StAux01.Pas
Serial Communication	StRem1.Pas
Turbo Pascal Run Time library modification	SE.ASM

ROM Kernel function description

EEPROM code loading instruction

Memory map and linker configuration

Hardware I/0 and Interrupts map

Batch and make files

Required Software Packages

The following commercial software packages are required to generate executable software for ARC Star Tracker:

Software from Borland International:

Turbo Pascal Version 6.0*
Turbo Pascal Run Time Library
Turbo Assembler Version 2.0*
Turbo C++ Version 1.0*

Software from Paradigm Systems:

Locate Version 3.1

Remote testing requires the following packages:
From Borland International:

TDrem Version 2.1
Turbo Debugger Version 3.0*

From Paradigm Systems:

Turbo Profilter Version 1.0*

Remarks --

Software packages marked with asterisk (*) have not been purchased by contract.

```
                  ;===========================================================================
                  ;
                  ; V1.00 5/10/92 Initial release
                  ; V1.01 7/25/92 New EEPROM access control
                  ; V1.02 8/13/92 Program execution control
                  ; V2.00 9/30.29 Modified test
                  ; V0.80 11/25/92 for flight boards

0000              DATA    segment para public 'DATA'      ; 00000 - 07fff
0000              DATA    ends
0000              CODE    segment para public 'CODE'      ; ff000 - fffff
0000              CODE    ends

                  ;       _______________
                  ;       Hardware equates
                  ;       _______________

      = 0200      WREE      equ    0200h
      = 1000      STOP      equ    1000h
      = 1100      IC00      equ    1100h
      = 1102      IC01      equ    1102h
      = 1200      PPI00     equ    1200h
      = 1202      PPI01     equ    1202h
      = 1204      PPI02     equ    1204h
      = 1206      PPI03     equ    1206h
      = 1300      UART00    equ    1300h
      = 1302      UART01    equ    1302h
      = 1400      ADCP      equ    1400h
      = 1500      PIT00     equ    1500h
      = 1502      PIT01     equ    1502h
      = 1504      PIT02     equ    1504h
      = 1506      PIT03     equ    1506h
      = 1600      WATCHD    equ    1600h
      = 1700      WGEN      equ    1700h
      = 0000      RAM1      equ    0000h
      = 2000      RAM2      equ    2000h
      = F000      EEPROM1 equ      0F000h
      = D000      EEPROM2 equ      0D000h
      = FE00      TDstrt    equ    0FE00h
      = 55AA      Tpttrn    equ    55AAh
      = 0008      EEPn1     equ    16-8
      = 0002      EEPn2     equ    16-14
      = 0001      EEPn3     equ    16-15
      = 0000      TstOfsE1 equ     0
      = 0000      TstOfsE2 equ     0
      = FFF6      TstOfsE3 equ     0FFF6h
      = 004E      BndRD9   equ     78      ;98 for 15 MHz  ; 78 for 12MHz
      = 0027      BndRD2   equ     39      ;49 for 15 MHz  ; 39 for 12MHz
      = 0014      BndRD4   equ     20      ;24 for 15 MHz  ; 20 for 12MHz
      = 000A      BndRD8   equ     10      ;12 for 15 MHz  ; 10 for 12MHz
      = 0007      BndRD115 equ      7      ; 8 for 15 MHz  ;  7 for 12MHz ; NOT supported by UART

      = 0020      EOI      equ     20h             ; non-specific EOI
      = 000F      WREEEN   equ     0fh             ; write eeprom enable
```

```

      = FE00                PROT_ROM equ    0FE00h          ; start of protected kernel ROM
      = FFF6                KerAbv  equ     0FFF6h          ; Above BOOT code - first above

      = FD00                UsrAseg equ     0FD000h         ; Area not protected by hardware
      = 0FFC                UsrAofs equ     0FFCh           ; Last 4 bytes before protected seg FE00h

      = E000                Ghost1start     equ 0E000h      ; Start of EEPROM1 ghost
      = F000                Ghost1end       equ 0F000h      ; First above ghost

                            ;       ----------------
                            ;       Interrupt Equates
                            ;       ----------------

      = 0008                RXRDY_INT equ   8               ; int 8  - receiver ready
      = 0009                TXRDY_INT equ   9               ; int 9  - transmitter ready

      = 0010                GETCH_INT equ   10h             ; int 10h - get character
      = 0011                PUTCH_INT equ   11h             ; int 11h - put character
      = 0012                WRTEE_INT equ   12h             ; int 12h - write EEPROM
      = 0013                TXRDY_S_INT equ 13h             ; int 13h - soft version of TXRDY_INT

                            ;       --------------------------------
                            ;       Global memory - initialized to zero
                            ;       --------------------------------

 0000                       DATA    segment

                                        org     20h             ; interrupt vector table
 0020  10*(????????)        ramivt      dd      16 dup(?)       ; (address sensitive)

                                        org     00h
 0000  0100*(????????)      ram_intr    dd      256 dup(?)      ; interrupt vectors space



 0400  ????                 rx_inp      dw      ?               ; rcv q input pointer  (i == o empty)
 0402  ????                 rx_outp     dw      ?               ; rcv q output pointer (i-1 == o full)
 0404  ????                 tx_inp      dw      ?               ; xmt q input pointer
 0406  ????                 tx_outp     dw      ?               ; xmt q output pointer
 0408  ????                 pwrup_err   dw      ?               ; powerup test results (0 == normal)  cx
                                                                ; 0 - RAM1 err                 $1 16
                                                                ; 1 - RAM1 err                 $2 15
                                                                ; 2 - RAM1 byte access err     $4 14
                                                                ; 3 - RAM1 addressing err      $8 13
                                                                ; 4 - 8259 int ctrlr err      $10 12
                                                                ; 5 - 8255 parallel port err  $20 11
                                                                ; 6 - 8254 timer err          $40 10
                                                                ; 7 - 8251 uart err           $80  9
                                                                ; 8 - EEPROM1 write error    $100  8
                                                                ; 9 - DSR not active         $200  7
                                                                ;10 - RAM2 err               $400  6
                                                                ;11 - RAM2 err               $800  5
                                                                ;12 - RAM2 byte access err  $1000  4
```

```
                                                                 ;13 - RAM2 addressing err      $2000  3
                                                                 ;14 - EEPROM2 write error      $4000  2
                                                                 ;15 - EEPROM1 KERNEL access    $8000  1

040A  52*(??)           msgbuf          db      82 dup(?)        ; message text from console
045C  40*(????)         ee_buf          dw      64  dup(?)       ; eeprom write buffer

                                        org     500h
0500  0100*(??)         tx_queue        db      256 dup(?)       ; xmit char output queue (256-aligned)
0600  0100*(??)         rx_queue        db      256 dup(?)       ; rcv char input queue (256-aligned)

0700                    wree_ram        label   FAR              ; this space used for wree subroutine
0700  0100*(??)                         db      100h dup(?)

0800  0100*(????)                       dw      100h dup (?)     ; 512 bytes of stack space
0A00                    stack_L         label   WORD
0A00                    DATA    ends

                        ;==========================================================================
0000                    CODE    segment
                                assume  cs:CODE, ds:DATA, es:DATA

                        ;       --------------------------------
                        ;       Startup code - reset entry point
                        ;       --------------------------------

0000                    start   proc    FAR

                        ;       set up segment registers

0000  FA                        cli                              ; Disable external interrupts
0001  FC                        cld                              ; Clear direction flag si,di increase
0002  B8 0000s                  mov     ax, DATA
0005  8E D8                     mov     ds, ax
0007  8E D0                     mov     ss, ax
0009  8E C0                     mov     es, ax
000B  B8 0A00r                  mov     ax, offset stack_L
000E  8B E0                     mov     sp, ax
0010  33 ED                     xor     bp, bp                   ;bp=0 No errors detected yet

                        ;       test 0a - RAM1 test
0012  B8 0000           test0a: mov     ax, RAM1
0015  8E C0                     mov     es, ax
0017  B8 55AA                   mov     ax, Tpttrn
001A  B9 8000                   mov     cx, 8000h                ;32Kwords to be tested
001D  33 FF                     xor     di, di
001F  33 F6                     xor     si, si
0021  F3> AB                    rep     stosw                    ;write to all 32Kwords
0023  B9 8000                   mov     cx, 8000h
0026  F3> AF                    repe    scasw                    ;compare with what's in ax
0028  74 03                     je      test0b
002A  83 CD 01                  or      bp, 1

                        ;       test 0b - RAM2 test
```

```
0020  B8 2000            test0b: mov     ax, RAM2
0030  8E C0                      mov     es, ax
0032  B8 55AA                    mov     ax, Tpttrn
0035  B9 8000                    mov     cx, 8000h               ;32Kwords to be tested
0038  33 FF                      xor     di, di
003A  33 F6                      xor     si, si
003C  F3> AB                     rep     stosw                   ;write to all 32Kwords
003E  B9 8000                    mov     cx, 8000h
0041  F3> AF                     repe    scasw                   ;compare with what's in ax
0043  74 04                      je      test1a
0045  81 CD 0400                 or      bp, 400h

                         ;       test 1a - RAM1 test

0049  B8 0000            test1a: mov     ax, RAM1
004C  8E C0                      mov     es, ax
004E  B8 55AA                    mov     ax, Tpttrn
0051  F7 D0                      not     ax
0053  B9 8000                    mov     cx, 8000h               ;32Kwords to be tested
0056  33 FF                      xor     di, di
0058  33 F6                      xor     si, si
005A  F3> AB                     rep     stosw                   ;write to all 32Kwords
005C  B9 8000                    mov     cx, 8000h
005F  F3> AF                     repe    scasw                   ;compare with what's in ax
0061  74 03                      je      test1b
0063  83 CD 02                   or      bp, 2

                         ;       test 1b - RAM2 test

0066  B8 2000            test1b: mov     ax, RAM2
0069  8E C0                      mov     es, ax
006B  B8 55AA                    mov     ax, Tpttrn
006E  F7 D0                      not     ax
0070  B9 8000                    mov     cx, 8000h               ;32Kwords to be tested
0073  33 FF                      xor     di, di
0075  33 F6                      xor     si, si
0077  F3> AB                     rep     stosw                   ;write to all 32Kwords
0079  B9 8000                    mov     cx, 8000h
007C  F3> AF                     repe    scasw                   ;compare with what's in ax
007E  74 04                      je      test2a
0080  81 CD 0800                 or      bp, 800h

                         ;       test 2a - byte access test RAM1
0084  8C DB              test2a: mov     bx, ds                  ; preserve ds

0086  B8 0000                    mov     ax, RAM1
0089  8E D8                      mov     ds, ax
008B  B8 55AA                    mov     ax, Tpttrn
008E  F7 D0                      not     ax
0090  B9 8000                    mov     cx, 8000h
0093  33 F6                      xor     si, si
0095  8B D0                      mov     dx, ax
0097  AC                 bytesta:lodsb
0098  3A C2                      cmp     al, dl
```

```
009A  75 07                     jne     test2af
009C  AC                        lodsb
009D  3A C6                     cmp     al, dh
009F  E1 F6                     loope   bytesta
00A1  74 03                     je      test2b
00A3  83 CD 04         test2af: or       bp, 4

                       ;        test 2b - byte access test RAM2

00A6  B8 2000          test2b:  mov     ax, RAM2
00A9  8E D8                     mov     ds, ax
00AB  B8 55AA                   mov     ax, Tpttrn
00AE  F7 D0                     not     ax
00B0  B9 8000                   mov     cx, 8000h
00B3  33 F6                     xor     si, si
00B5  8B D0                     mov     dx, ax
00B7  AC               bytestb: lodsb
00B8  3A C2                     cmp     al, dl
00BA  75 07                     jne     test2bf
00BC  AC                        lodsb
00BD  3A C6                     cmp     al, dh
00BF  E1 F6                     loope   bytestb
00C1  74 04                     je      test3a
00C3  81 CD 1000       test2bf: or       bp, 1000h

00C7                   test3a:

                       ;        test 3a - address test RAM1

00C7  B8 0000                   mov     ax, RAM1
00CA  8E C0                     mov     es, ax
00CC  8E D8                     mov     ds, ax
00CE  B9 8000                   mov     cx, 8000h
00D1  33 FF                     xor     di, di
00D3  8B C7            test3b:  mov     ax, di
00D5  F7 D0                     not     ax
00D7  AB                        stosw
00D8  E2 F9                     loop    test3b

00DA  B9 8000                   mov     cx, 8000h
00DD  33 F6                     xor     si, si
00DF  8B D6            test3c:  mov     dx, si
00E1  AD                        lodsw
00E2  F7 D0                     not     ax
00E4  3B C2                     cmp     ax, dx
00E6  E1 F7                     loope   test3c
00E8  74 03                     je      test3d
00EA  83 CD 08                  or      bp, 8

                       ;        test 3d - address test RAM2

00ED  B8 2000          test3d:  mov     ax, RAM2
00F0  8E C0                     mov     es, ax
00F2  8E D8                     mov     ds, ax
```

```
00F4  B9 8000                  mov    cx, 8000h
00F7  33 FF                    xor    di, di
00F9  8B C7           test3e: mov    ax, di
00FB  F7 D0                    not    ax
00FD  AB                       stosw
00FE  E2 F9                    loop   test3e

0100  B9 8000                  mov    cx, 8000h
0103  33 F6                    xor    si, si
0105  8B D6           test3f: mov    dx, si
0107  AD                       lodsw
0108  F7 D0                    not    ax
010A  3B C2                    cmp    ax, dx
010C  E1 F7                    loope  test3f
010E  74 04                    je     test3g
0110  81 CD 2000               or     bp, 2000h

0114  8E DB           test3g: mov    ds, bx              ; restore ds

                      ;        Leave memory zeroed

0116  B8 0000                  mov    ax, RAM1
0119  8E C0                    mov    es, ax
011B  B9 8000                  mov    cx, 8000h
011E  33 C0                    xor    ax, ax
0120  8B F8                    mov    di, ax
0122  F3> AB                   rep    stosw

0124  B8 2000                  mov    ax, RAM2
0127  8E C0                    mov    es, ax
0129  B9 8000                  mov    cx, 8000h
012C  33 C0                    xor    ax, ax
012E  8B F8                    mov    di, ax
0130  F3> AB                   rep    stosw

                      ; restore es

0132  B8 0000s                 mov    ax, DATA
0135  8E C0                    mov    es, ax

                      ;        Copy interrupt vector table from ROM to RAM

0137  1E                       push   ds
0138  BE 07E2r                 mov    si, OFFSET romivt
013B  0E                       push   cs
013C  1F                       pop    ds
013D  BF 0020r                 mov    di, OFFSET ramivt
0140  B9 0020                  mov    cx, 32
0143  F3> A5                   rep    movsw
0145  1F                       pop    ds

                      ;        test 4 - 8259 interrupt controller test

0146  BA 1100                  mov    dx, IC00      ; 8259 port 0
```

```
0149  B0 13                  mov     al, 13h         ; ICW1: edge trig, sngl, icw4
014B  EE                     out     dx, al
014C  BA 1102                mov     dx, IC01        ; 8259 port 1
014F  B0 08                  mov     al, 8           ; ICW2: t=8 (addr = 20h)
0151  EE                     out     dx, al
0152  B0 01                  mov     al, 1           ; ICW4: no sfnm, non buffered, normal eoi, 86
0154  EE                     out     dx, al
0155  EC                     in      al, dx          ; read mask reg
0156  0A C0                  or      al, al          ; should be zero
0158  74 03                  jz      test4a
015A  83 CD 10               or      bp, 10h
015D  B0 FF          test4a: mov     al, 0ffh        ; now set all mask bits
015F  EE                     out     dx, al
0160  32 C0                  xor     al, al
0162  EC                     in      al, dx
0163  3C FF                  cmp     al, 0ffh
0165  74 03                  je      test5
0167  83 CD 10               or      bp, 10h

                     ;       test 5 - 8255 PPI test

016A  BA 1206        test5:  mov     dx, PPI03
016D  B0 92                  mov     al, 92h         ; A,B mode 0 input; C mode 0 output
016F  EE                     out     dx, al
0170  32 C0                  xor     al, al
0172  EC                     in      al, dx          ; read back mode
0173  3C 92                  cmp     al, 92h         ; did it stick?
0175  74 03                  je      test6
0177  83 CD 20               or      bp, 20h

                     ;       test 6 - 8254 PIT test

017A  BA 1506        test6:  mov     dx, PIT03       ; port 3 == control port
017D  B0 30                  mov     al, 30h         ; counter 0, lsb+msb, mode 0
017F  EE                     out     dx, al
0180  B0 74                  mov     al, 74h         ; counter 1, lsb+msb, mode 2
0182  EE                     out     dx, al
0183  BA 1502                mov     dx, PIT01
0186  B8 FFFF                mov     ax, 0ffffh      ; count = 65535 (input 0.66MHz, output 10.2Hz)
0189  EE                     out     dx, al
018A  8A C4                  mov     al, ah
018C  EE                     out     dx, al
018D  B0 B6                  mov     al, 0B6h        ; counter 2, lsb+msb, mode 3
018F  BA 1506                mov     dx, PIT03       ; output to control port
0192  EE                     out     dx, al
0193  BA 1504                mov     dx, PIT02
0196  B8 004E                mov     ax, BndRD9      ; 9600kHz baud rate
0199  EE                     out     dx, al          ; set lo byte
019A  8A C4                  mov     al, ah
019C  EE                     out     dx, al          ; set hi byte
                                                     ; count = 96 for 15MHz
                                                     ; count = 32 for 5MHz
                                                     ; count = 26 (input 4MHz, output 153.846KHz)
019D  EC                     in      al, dx          ; now test if counter 2 is counting
```

```
019E  8A E0                        mov     ah, al
01A0  EC                           in      al, dx
01A1  3A E0                        cmp     ah, al
01A3  75 03                        jne     test7           ; counts differ, it's working
01A5  83 CD 40                     or      bp, 40h
                          ;        test 7 - 8251 UART test
01A8  BA 1302             test7:   mov     dx, UART01
01AB  32 C0                        xor     al, al
01AD  EE                           out     dx, al
01AE  EE                           out     dx, al
01AF  EE                           out     dx, al
01B0  B0 40                        mov     al, 40h         ; internal reset
01B2  EE                           out     dx, al
01B3  B0 6E                        mov     al, 6eh         ; 16x (9600 baud), 8 bits, no parity 1 stop bit
                          ;        mov     al, 6Dh         ; 1x, 8 bits, no parity, 1 stop
01B5  EE                           out     dx, al
01B6  B0 37                        mov     al, 37h         ; rts, err reset, rcv ena, dtr, xmt ena
01B8  EE                           out     dx, al
01B9  EC                           in      al, dx          ; get status reg
01BA  A8 38                        test    al, 38h         ; test for FE, OE, PE
01BC  74 04                        jz      test7a          ; jmp of none are set
01BE  81 CD 0080                   or      bp, 80h         ; one or more bits were set
01C2  A8 80               test7a:  test    al, 80h         ; test for DSR bit
01C4  75 04                        jnz     test7x          ; if nz, DSR active
01C6  81 CD 0200                   or      bp, 200h        ; set to show DSR inactive
01CA  24 05               test7x:  and     al, 5           ; test for TxE, TxRdy
01CC  3C 05                        cmp     al, 5
01CE  74 04                        je      test7b          ; jmp if all set
01D0  81 CD 0080                   or      bp, 80h         ; one or more bits were not set
01D4  BA 1300             test7b:  mov     dx, UART00
01D7  F7 C5 0200                   test    bp, 200h        ; is DSR active?
01DB  75 0F                        jnz     test7c          ; n -- don't send char (not debug mode)
01DD  B0 0D                        mov     al, 0dh         ; send out <cr>
01DF  EE                           out     dx, al
01E0  BA 1302                      mov     dx, UART01      ; now read status
01E3  EC                           in      al, dx
01E4  A8 04                        test    al, 4           ; test for xmtr rdy
01E6  74 04                        jz      test7c          ; ..should not be ready at this point
01E8  81 CD 0080                   or      bp, 80h
01EC                      test7c:
                          ;        Copy EEPROM access routine into RAM
01EC  1E                           push    ds
01ED  06                           push    es
01EE  BF 0700r                     mov     di, OFFSET wree_ram     ; RAM area ds=RAm1
01F1  BE 0743r                     mov     si, OFFSET wree_rom     ; ROM area es=EEPROM1
01F4  B9 009E 90                   mov     cx, wree_len            ; length of code
01F8  8C C8                        mov     ax, cs
01FA  8E D8                        mov     ds, ax
01FC  F3> A4                       rep     movsb                   ; ds:si --> es:di
01FE  07                           pop     es
```

```
0 1FF  1F                        pop     ds

                         ;       EEPROM1 access test

0200  1E                 test8:  push    ds
0201  06                         push    es
0202  B8 0000                    mov     ax, RAM1
0205  8E D8                      mov     ds, ax
0207  B8 F000                    mov     ax, EEPROM1
020A  8E C0                      mov     es, ax                ; now ds-->RAM, es-->EEPROM

020C  BF FFF6                    mov     di, TstOfsE3          ; test word in Kernel above BOOT
020F  BE 045Cr                   mov     si, OFFSET ee_buf
0212  26: 8B 05                  mov     ax, es:[di]           ; get pattern word
0215  F7 D0                      not     ax                    ; now invert it
0217  89 04                      mov     ds:[si], ax           ; place in RAM buffer
0219  B9 0001                    mov     cx, 1                 ; copy 1 word
021C  CD 12                      int     WRTEE_INT             ; store in EEPROM

021E  0B C0                      or      ax, ax                ; test if stored
0220  74 04                      je      test8a
0222  81 CD 8000                 or      bp, 8000h             ; EEPROM KERNEL access

0226  B8 F000            test8a: mov     ax, EEPROM1
0229  8E C0                      mov     es, ax
022B  BF 0000                    mov     di, TstOfsE1          ; Test location in EEPROM1
022E  BE 045Cr                   mov     si, OFFSET ee_buf
0231  26: 8B 05                  mov     ax, es:[di]           ; get pattern word
0234  50                         push    ax                    ; save it
0235  F7 D0                      not     ax                    ; now invert it
0237  89 04                      mov     ds:[si], ax           ; place in RAM buffer
0239  B9 0001                    mov     cx, 1                 ; copy 1 word
023C  CD 12                      int     WRTEE_INT             ; store in EEPROM

023E  0B C0                      or      ax, ax                ; test if stored
0240  74 04                      je      test8b
0242  81 CD 0100                 or      bp, 100h              ; EEPROM1 access

0246  B8 F000            test8b: mov     ax, EEPROM1           ; EEPROM1 access
0249  8E C0                      mov     es, ax
024B  BF 0000                    mov     di, TstOfsE1          ; location to restore
024E  58                         pop     ax                    ; get old value
024F  89 04                      mov     ds:[si], ax           ; place old val in RAM buffer
0251  B9 0001                    mov     cx, 1
0254  CD 12                      int     WRTEE_INT             ; RESTORE EEPROM

0256  B8 D000            test8c: mov     ax, EEPROM2
0259  8E C0                      mov     es, ax
025B  BF 0000                    mov     di, TstOfsE2          ; Test location in EEPROM2
025E  BE 045Cr                   mov     si, OFFSET ee_buf
0261  26: 8B 05                  mov     ax, es:[di]           ; get pattern word
0264  50                         push    ax                    ; save it
0265  F7 D0                      not     ax                    ; now invert it
0267  89 04                      mov     ds:[si], ax           ; place in RAM buffer
```

```
0269  B9 0001                     mov     cx, 1                   ; copy 1 word
026C  CD 12                       int     WRTEE_INT               ; store in EEPROM

026E  0B C0                       or      ax, ax                  ; test if stored
0270  74 04                       je      test8d
0272  81 CD 4000                  or      bp, 4000h               ; EEPROM2 access

0276  B8 D000             test8d: mov     ax, EEPROM2
0279  8E C0                       mov     es, ax
027B  BF 0000                     mov     di, TstOfsE2            ; location to restore
027E  58                          pop     ax                      ; get old value
027F  89 04                       mov     ds:[si], ax             ; place old val in RAM buffer
0281  B9 0001                     mov     cx, 1
0284  CD 12                       int     WRTEE_INT               ; RESTORE EEPROM

0286  07                          pop     es
0287  1F                          pop     ds

                          ;       Set up pointers and variables

0288  B8 0600r                    mov     ax, OFFSET rx_queue     ; init rcv queue pointers
028B  A3 0400r                    mov     rx_inp, ax              ; (to empty state)
028E  A3 0402r                    mov     rx_outp, ax
0291  B8 0500r                    mov     ax, OFFSET tx_queue     ; init xmt queue pointers
0294  A3 0404r                    mov     tx_inp, ax              ; (to empty state)
0297  A3 0406r                    mov     tx_outp, ax
029A  89 2E 0408r                 mov     pwrup_err, bp           ; store powerup test results

                          ;       Enable interrupts

029E  B0 E4                       mov     al, 0e4h                ; unmask rxrdy, txrdy, timebase, timer
02A0  BA 1102                     mov     dx, IC01                ; output to 8259
02A3  EE                          out     dx, al
02A4  FB                          sti

                          ;       Wait for 'esc' character from serial port...
                          ;       if it arrives, user wants monitor control.
                          ;       If it does not, transfer to user program.

02A5  F7 C5 0200                  test    bp, 200h                ; is DSR active (debug port)?
                                  ;jz      monitr                  ; y -- go straight to monitor
02A9  EB 10 90                    jmp     monitr
02AC  2B C9                       sub     cx, cx                  ; loop counter
02AE  CD 10               waitec: int     GETCH_INT               ; get char from serial port
02B0  0A E4                       or      ah, ah                  ; did we get one?
02B2  E0 FA                       loopne  waitec                  ; jmp if not, looping
02B4  3C 1B                       cmp     al, 27                  ; did we get an esc char?
02B6  74 03                       je      monitr                  ; jmp if so, enter monitor
02B8  E9 0172                     jmp     gouser                  ; go to user program

                          ;       Write identification message

02BB  BA 0822r            monitr: mov     dx, OFFSET idmsg
02BE  0E                          push    cs
```

```
02BF  07                        pop     es
02C0  E8 00CE                   call    putmsg
02C3  BA 0841r                  mov     dx, OFFSET idmsg2
02C6  0E                        push    cs
02C7  07                        pop     es
02C8  E8 00C6                   call    putmsg

                        ;       Print diagnostic results

02CB  8B 2E 0408r               mov     bp, pwrup_err
02CF  B9 0010                   mov     cx, 16                  ; test 16 bits
02D2  BF 0001                   mov     di, 1
02D5  BB 0A02r                  mov     bx, OFFSET bitmsg
02D8  0E                        push    cs
02D9  07                        pop     es
02DA  2E: 8B 17         diag00: mov     dx, cs:[bx]             ; output test name
02DD  E8 00B1                   call    putmsg
02E0  85 EF                     test    bp, di                  ; check pass/fail bit
02E2  74 2F                     jz      diag01                  ; z -- passed
02E4  83 F9 08                  cmp     cx, EEPn1
02E7  75 08                     jne     diag03
02E9  BA 08A7r                  mov     dx, OFFSET denn
02EC  E8 00A2                   call    putmsg
02EF  EB 28                     jmp     SHORT diag02
02F1  83 F9 02          diag03: cmp     cx, EEPn2
02F4  75 08                     jne     diag04
02F6  BA 08A7r                  mov     dx, OFFSET denn
02F9  E8 0095                   call    putmsg
02FC  EB 1B                     jmp     SHORT diag02
02FE  83 F9 01          diag04: cmp     cx, EEPn3
0301  75 08                     jne     diagf
0303  BA 08A7r                  mov     dx, OFFSET denn
0306  E8 0088                   call    putmsg
0309  EB 0E                     jmp     SHORT diag02
030B  BA 089Cr          diagf:  mov     dx, OFFSET fail
030E  E8 0080                   call    putmsg                  ; print fail message
0311  EB 06                     jmp     SHORT diag02            ; continue with next test
0313  BA 0891r          diag01: mov     dx, OFFSET pass
0316  E8 0078                   call    putmsg                  ; print pass message
0319  83 C3 02          diag02: add     bx, 2                   ; continue
031C  D1 E7                     shl     di, 1
031E  E2 BA                     loop    diag00


                        ;       Enter infinite wait

0320                    xxx:
0320  BA 087Fr                  mov     dx, OFFSET prompt
0323  0E                        push    cs
0324  07                        pop     es
0325  E8 0069                   call    putmsg
0328  BA 040Ar                  mov     dx, OFFSET msgbuf
032B  B9 0050                   mov     cx, 80
032E  E8 0007                   call    getmsg
```

```
0331  E8 0090                       call    process
0334  EB EA                         jmp     xxx

0336                        start   endp

                            ;       ----------------------------
                            ;       Read message from serial port
                            ;       ----------------------------
                            ;       Call with:
                            ;               ds:dx --> buffer to receive string
                            ;               cx =      maximum size of receive buffer (min 2)
                            ;       Returns:
                            ;               ds:dx --> unchanged
                            ;               cx =      actual number of chars read (incl CR)

0336  90                            nop
0337  90                            nop
0338                        getmsg  proc    NEAR
                                    assume  ds:DATA, es:nothing

0338  56                            push    si
0339  57                            push    di
033A  52                            push    dx
033B  1E                            push    ds
033C  06                            push    es

033D  8C D8                         mov     ax, ds
033F  8E C0                         mov     es, ax
0341  8B FA                         mov     di, dx                  ; now es:di --> buffer

0343  B8 0000s                      mov     ax, DATA
0346  8E D8                         mov     ds, ax
0348  8B 1E 0402r                   mov     bx, rx_outp             ; now ds:bx --> rcv queue

034C  49                            dec     cx                      ; leave one extra for terminating NUL

034D  3B 1E 0400r           getm01: cmp     bx, rx_inp              ; queue empty?
0351  74 FA                         je      getm01                  ; y -- wait for char
0353  8A 07                         mov     al, [bx]
0355  FE C3                         inc     bl
0357  3C 00                         cmp     al, 0                   ; NUL -- turbo debugger packet
0359  75 03                         jne     getm02
035B  E9 01FD                       jmp     xdeb                    ; go transfer to debugger
035E  3C 0D                 getm02: cmp     al, 0dh                 ; carriage return?
0360  74 1D                         je      getm10                  ; y -- end of message
0362  3C 7F                         cmp     al, 7fh                 ; delete key?
0364  75 14                         jne     getm04                  ; n -- normal char
0366  3B FA                         cmp     di, dx                  ; at beginning of buffer already?
0368  74 E3                         je      getm01                  ; jmp if so
036A  4F                            dec     di                      ; back up one char
036B  49                            dec     cx
036C  B0 08                         mov     al, 8                   ; send bs-s-bs sequence
036E  CD 11                         int     PUTCH_INT
0370  B0 20                         mov     al, ' '
```

```
0372  CD 11                        int     PUTCH_INT
0374  B0 08                        mov     al, 8
0376  CD 11                        int     PUTCH_INT
0378  EB D3                        jmp     getm01
037A  AA                   getm04: stosb                          ; store in es:[di++]
037B  CD 11                        int     PUTCH_INT              ; echo it
037D  E2 CE                getm05: loop    getm01

037F  AA                   getm10: stosb                          ; save CR $0d
0380  89 1E 0402r                  mov     rx_outp, bx            ; save new output pointer
0384  8B CF                        mov     cx, di                 ; calculate number of rcv'd chars
0386  2B CA                        sub     cx, dx
0388  32 C0                        xor     al, al
038A  AA                           stosb                          ; store terminating NUL
038B  07                           pop     es
038C  1F                           pop     ds
038D  5A                           pop     dx
038E  5F                           pop     di
038F  5E                           pop     si
0390  C3                           ret
0391                       getmsg  endp


                           ;       ----------------------------
                           ;       Output message to serial port
                           ;       ----------------------------
                           ;       Call with:
                           ;               es:dx --> null-terminated string
                           ;
                           ;       All regs except ax preserved
                           ;
0391                       putmsg  proc    NEAR
                                   assume  es:DATA, ds:nothing

0391  53                           push    bx
0392  56                           push    si
0393  1E                           push    ds
0394  06                           push    es
0395  8C C0                        mov     ax, es
0397  8E D8                        mov     ds, ax
0399  8B F2                        mov     si, dx                 ; now ds:si --> message
039B  B8 0000s                     mov     ax, DATA
039E  8E C0                        mov     es, ax
03A0  26: 8B 1E 0404r              mov     bx, tx_inp             ; now es:bx --> transmit queue
03A5  AC                   msgo01: lodsb
03A6  0A C0                        or      al, al
03A8  74 0E                        jz      msgo04
03AA  26: 88 07                    mov     es:[bx], al
03AD  FE C3                        inc     bl
03AF  26: 3B 1E 0406r      msgo02: cmp     bx, tx_outp            ; queue full?
03B4  74 F9                        je      msgo02                 ; y -- wait here until slot avail
03B6  EB ED                        jmp     msgo01

03B8  26: 89 1E 0404r      msgo04: mov     tx_inp, bx
```

```
03BD CD 13               int     TXRDY_S_INT
03BF 07        msgo06:   pop     es
03C0 1F                  pop     ds
03C1 5E                  pop     si
03C2 5B                  pop     bx
03C3 C3                  ret
03C4           putmsg    endp

                         assume  ds:DATA, es:DATA

               ;         ---------------------
               ;         Process command message
               ;         ---------------------
03C4           process proc      NEAR

03C4 B0 0D               mov     al, 0dh
03C6 CD 11               int     PUTCH_INT
03C8 B0 0A               mov     al, 0ah
03CA CD 11               int     PUTCH_INT

03CC 8B F2               mov     si, dx
03CE AC                  lodsb                      ; get cmd char
03CF 0C 20               or      al, ' '            ; convert to lower case

03D1 3C 34     pr040:    cmp     al, '4'
03D3 75 04               jne     pr042
03D5 E8 0192             call    Br40               ; change Baud rate to 40kHz
03D8 C3                  ret

03D9 3C 39     pr042:    cmp     al, '9'
03DB 75 04               jne     pr044
03DD E8 01C3             call    Br9                ; change Baud rate to 9.6kHz
03E0 C3                  ret

03E1 3C 31     pr044:    cmp     al, '1'
03E3 75 04               jne     pr046
03E5 E8 01CE             call    Br115              ; change Baud rate to 115kHz
03E8 C3                  ret

03E9 3C 32     pr046:    cmp     al, '2'
03EB 75 04               jne     pr048
03ED E8 018D             call    Br2                ; change Baud rate to 19.2kHz
03F0 C3                  ret

03F1 3C 38     pr048:    cmp     al, '8'
03F3 75 04               jne     pr050
03F5 E8 0198             call    Br8                ; change Baud rate to 80kHz
03F8 C3                  ret

03F9 3C 70     pr050:    cmp     al, 'p'            ; p -- print diagnostics
03FB 75 03               jne     pr052
03FD E9 FEBB             jmp     monitr             ; restart monitor

0400 3C 72     pr052:    cmp     al,'r'             ; Hard Reset from EEPROM1
```

```
0402  75 05                     jne     pr054
                                IDEAL
0404  EA F000FFF0               JMP     FAR 0F000h:0FFF0h
                                MASM
0409  3C 73             pr054:  cmp     al, 's'
040B  75 05                     jne     pr060
                                IDEAL
040D  EA D000FFF0               JMP     FAR 0D000h:0FFF0h
                                MASM
0412  3C 77             pr060:  cmp     al, 'w'              ; w -- write user prog address
0414  75 0D                     jne     pr065
0416  E8 01B0                   call    hex2bin              ; get segment
0419  8B D8                     mov     bx, ax
041B  46                        inc     si                   ; skip term char
041C  E8 01AA                   call    hex2bin              ; get offset
041F  E8 0072                   call    write_useradd        ; invoke routine
0422  C3                        ret
0423  3C 67             pr065:  cmp     al, 'g'              ; g -- go to user program
0425  75 29                     jne     pr070
0427  8A 04                     mov     al, [si]             ; get next cmd char
0429  3C 0D                     cmp     al, 0dh              ; is it CR (no args)?
042B  75 17                     jne     pr066                ; jmp if not
042D  06                gouser: push    es
042E  56                        push    si
042F  B8 FD00                   mov     ax, UsrAseg
0432  8E C0                     mov     es, ax
0434  BE 0FFC                   mov     si, UsrAofs          ; es:si points to useradress ofset
0437  26: 8B 04                 mov     ax, es:si            ; load user adress ofset
043A  46                        inc     si
043B  46                        inc     si                   ; segment is 2 bytes after ofset
043C  26: 8B 1C                 mov     bx, es:si            ; load user adress segment
043F  5E                        pop     si
0440  07                        pop     es
0441  EB 67 90                  jmp     go_useradd           ; use eeprom user entry point
0444  E8 0182           pr066:  call    hex2bin              ; get next number (seg)
0447  8B D8                     mov     bx, ax
0449  46                        inc     si                   ; skip ':'
044A  E8 017C                   call    hex2bin              ; get next number (off)
044D  EB 5B 90                  jmp     go_useradd
0450  3C 64             pr070:  cmp     al, 'd'
0452  75 1D                     jne     pr080
0454  E8 0172                   call    hex2bin
0457  50                        push    ax                   ; save seg
0458  8A 04                     mov     al, [si]             ; get next char
045A  3C 0D                     cmp     al, 0dh              ; is it CR (no length)
045C  75 06                     jne     pr072
045E  B9 0008                   mov     cx, 8                ; length := 8 if not speciffied
0461  EB 07 90                  jmp     pr074
0464  46                pr072:  inc     si                   ; skip term char
```

```
0465  E8 0161                     call    hex2bin                 ; get length
0468  8B C8                       mov     cx, ax
046A  58                  pr074:  pop     ax
046B  8E C0                       mov     es, ax
046D  E8 003D                     call    dumpmem
0470  C3                          ret

0471  3C 6C               pr080:  cmp     al, 'l'
0473  75 18                       jne     pr099
0475  E8 0151                     call    hex2bin         ; convert next number to binary
0478  8E C0                       mov     es, ax          ; keep segment
047A  AC                          lodsb
047B  3C 0D                       cmp     al, 0dh         ; did cr terminate line?
047D  75 05                       jne     pr082           ; jmp if not, get byte count
047F  B9 0080                     mov     cx, 128         ; no count specified, use 128
0482  EB 05                       jmp     SHORT pr084
0484  E8 0142             pr082:  call    hex2bin         ; convert next number to binary
0487  8B C8                       mov     cx, ax
0489  E8 006C             pr084:  call    progload        ; xfer to program loader
048C  C3                          ret

048D  0E                  pr099:  push    cs
048E  07                          pop     es
048F  B0 3F                       mov     al, '?'
0491  CD 11                       int     PUTCH_INT
0493  C3                          ret
0494                      process endp

                          ;       ----------------------------------------
                          ;       Write user entry point address to EEPROM
                          ;       ----------------------------------------
                          ;       bx = segment
                          ;       ax = offset
                          ;
0494                      write_useradd proc NEAR
0494  BE 045Cr                    mov     si, OFFSET ee_buf               ; ds:si --> eeprom write buf
0497  89 04                       mov     [si], ax                        ; copy segment and offset
0499  89 5C 02                    mov     [si+2], bx
049C  B8 FD00                     mov     ax, UsrAseg
049F  8E C0                       mov     es, ax
04A1  BF 0FFC                     mov     di, UsrAofs                     ; es:di --> user addr FD00:0FFC
04A4  B9 0002                     mov     cx, 2                           ; write 2 words
04A7  CD 12                       int     WRTEE_INT
04A9  C3                          ret
04AA                      write_useradd endp

                          ;       --------------------------------
                          ;       Transfer control to user program
                          ;       --------------------------------
                          ;       bx = segment
                          ;       ax = offset
                          ;
04AA                      go_useradd proc FAR
04AA  53                          push    bx
```

```
04AB  50                        push    ax
04AC  CB                        ret
04AD                    go_useradd endp


                        ;       ------------
                        ;       Memory Dumper
                        ;       ------------

04AD                    dumpmem proc    NEAR
04AD  8C C0                     mov     ax, es
04AF  51                dmp001: push    cx
04B0  BE 040Ar                  mov     si, OFFSET msgbuf
04B3  E8 014F                   call    bin2hex
04B6  06                        push    es
04B7  1E                        push    ds
04B8  07                        pop     es
04B9  8B D6                     mov     dx, si
04BB  E8 FED3                   call    putmsg
04BE  07                        pop     es
04BF  B0 3A                     mov     al, ':'
04C1  32 E4                     xor     ah, ah
04C3  CD 11                     int     PUTCH_INT
04C5  B0 20                     mov     al, ' '
04C7  CD 11                     int     PUTCH_INT
04C9  B9 0008                   mov     cx, 8
04CC  33 DB                     xor     bx, bx
04CE  26: 8B 07         dmp010: mov     ax, es:[bx]
04D1  E8 0131                   call    bin2hex
04D4  06                        push    es
04D5  1E                        push    ds
04D6  07                        pop     es
04D7  E8 FEB7                   call    putmsg
04DA  07                        pop     es
04DB  B0 20                     mov     al, ' '
04DD  32 E4                     xor     ah, ah
04DF  CD 11                     int     PUTCH_INT
04E1  83 C3 02                  add     bx, 2
04E4  E2 E8                     loop    dmp010
04E6  B8 000D                   mov     ax, 0dh
04E9  CD 11                     int     PUTCH_INT
04EB  B0 0A                     mov     al, 0ah
04ED  CD 11                     int     PUTCH_INT
04EF  8C C0                     mov     ax, es
04F1  40                        inc     ax
04F2  8E C0                     mov     es, ax
04F4  59                        pop     cx
04F5  E2 B8                     loop    dmp001
04F7  C3                        ret
04F8                    dumpmem endp

                        ;       ------------
                        ;       Program Loader
                        ;       ------------
```

```
                              ;       Input:
                              ;               es  --> segment address of start of program load
                              ;               cx = byte count (0-128)
                              ;
                              ;       Receives binary data from serial port input, places in
                              ;       temporary buffer in RAM. Invokes EEPROM write routine.
                              ;       Issues error message to console if write error occurs.

04F8                          progload proc   NEAR

04F8  06                              push    es                      ; save segment
04F9  51                              push    cx                      ; save count
04FA  D1 E9                           shr     cx, 1                   ; convert bytes to words
04FC  E3 23                           jcxz    prl15                   ; word count 0 -- skip word loop
04FE  B8 0000s                        mov     ax, DATA
0501  8E C0                           mov     es, ax
0503  BF 045Cr                        mov     di, OFFSET ee_buf       ; es:di --> ee_buf
0506  CD 10                   prl10:  int     GETCH_INT               ; get low byte
0508  0B C0                           or      ax, ax
050A  7C FA                           jl      prl10
050C  8A D8                           mov     bl, al
050E  CD 10                   prl12:  int     GETCH_INT               ; get high byte
0510  0B C0                           or      ax, ax
0512  7C FA                           jl      prl12
0514  8A F8                           mov     bh, al                  ; assemble bytes
0516  8B C3                           mov     ax, bx
0518  81 FF 04DCr                     cmp     di, OFFSET ee_buf+128   ; written 64 words yet?
051C  73 01                           jae     prl14                   ; jmp if so, skip buffering
051E  AB                              stosw                           ; put in buffer
051F  E2 E5                   prl14:  loop    prl10
0521  59                      prl15:  pop     cx                      ; restore count
0522  07                              pop     es                      ; restore segment
0523  F7 C1 0001                      test    cx, 1                   ; byte count odd?
0527  74 0E                           jz      prl17                   ; jmp if not
0529  CD 10                   prl16:  int     GETCH_INT               ; pick up odd char
052B  0B C0                           or      ax, ax
052D  7C FA                           jl      prl16
052F  81 FF 04DCr                     cmp     di, OFFSET ee_buf+128
0533  73 02                           jae     prl17
0535  AB                              stosw                           ; store odd char
0536  41                              inc     cx                      ; and make sure block count is even
0537  33 FF                   prl17:  xor     di, di                  ; es:di --> eeprom load address
0539  D1 E9                           shr     cx, 1                   ; convert bytes to words
053B  B8 0000s                        mov     ax, DATA
053E  8E D8                           mov     ds, ax
0540  BE 045Cr                        mov     si, OFFSET ee_buf       ; ds:si --> ee_buf
0543  CD 12                           int     WRTEE_INT               ; write to eeprom
0545  0B C0                           or      ax, ax                  ; check return code
0547  74 11                           jz      prl99                   ; zero, successful
0549  50                              push    ax                      ; save return code
054A  BA 0886r                        mov     dx, OFFSET wrterr       ; print err message
054D  0E                              push    cs
054E  07                              pop     es
054F  E8 FE3F                         call    putmsg
```

```
0552  58                  pop     ax                ; retrieve error code
0553  F7 D8               neg     ax                ; turn into positive number
0555  05 0030             add     ax, '0'           ; convert to ascii
0558  CD 11               int     PUTCH_INT         ; append to err message
055A  C3         prl99:   ret

055B             progload endp

                 ;        ------------------------------------
                 ;        Transfer to Turbo Debugger Remote Kernel
                 ;        ------------------------------------

055B             xdeb     proc    NEAR
055B  FA                  cli
055C  B0 FF               mov     al, 0ffh          ; mask all interrupts
055E  BA 1102             mov     dx, IC01
0561  EE                  out     dx, al



0562  B8 FE00             mov     ax, TDstrt        ; xfer to Turbo Debug kernel
                                                    ; 0fe00:0 (segment)
0565  50                  push    ax
0566  33 C0               xor     ax, ax            ;   (offset)
0568  50                  push    ax
                 ;        sti
0569             temp     proc    FAR
0569  CB                  ret                       ; ..by doing a far return
056A             temp     endp
056A             xdeb     endp

056A             Br40     proc NEAR

056A  FA                  cli
056B  BA 1506             mov     dx, PIT03         ; Baud gen. control port
056E  B0 B6               mov     al, 0B6h          ; cntr2, lsb+msb, mode 3
0570  EE                  out     dx, al
0571  BA 1504             mov     dx, PIT02         ; cntr2 data
0574  B8 0014             mov     ax, BndRD4        ; 40kHz Baud rate
0577  EE                  out     dx, al            ; set low byte
0578  8A C4               mov     al, ah
057A  EE                  out     dx, al            ; set hi byte
057B  FB                  sti

057C  C3                  ret
057D             Br40     endp

057D             Br2      proc NEAR

057D  FA                  cli
057E  BA 1506             mov     dx, PIT03         ; Baud gen. control port
0581  B0 B6               mov     al, 0B6h          ; cntr2, lsb+msb, mode 3
0583  EE                  out     dx, al
0584  BA 1504             mov     dx, PIT02         ; cntr2 data
```

```
0587  B8 0027                   mov     ax, BndRD2              ; 19.2kHz Baud rate
058A  EE                        out     dx, al                  ; set low byte
058B  8A C4                     mov     al, ah
058D  EE                        out     dx, al                  ; set hi byte
058E  FB                        sti

058F  C3                        ret
0590                    Br2     endp

0590                    Br8     proc NEAR

0590  FA                        cli
0591  BA 1506                   mov     dx, PIT03               ; Baud gen. control port
0594  B0 B6                     mov     al, 0B6h                ; cntr2, lsb+msb, mode 3
0596  EE                        out     dx, al
0597  BA 1504                   mov     dx, PIT02               ; cntr2 data
059A  B8 000A                   mov     ax, BndRD8              ; 80kHz Baud rate
059D  EE                        out     dx, al                  ; set low byte
059E  8A C4                     mov     al, ah
05A0  EE                        out     dx, al                  ; set hi byte
05A1  FB                        sti

05A2  C3                        ret
05A3                    Br8     endp

05A3                    Br9     proc NEAR

05A3  FA                        cli
05A4  BA 1506                   mov     dx, PIT03               ; Baud gen. control port
05A7  B0 B6                     mov     al, 0B6h                ; cntr2, lsb+msb, mode 3
05A9  EE                        out     dx, al
05AA  BA 1504                   mov     dx, PIT02               ; cntr2 data
05AD  B8 004E                   mov     ax, BndRD9              ; 9.6kHz Baud rate
05B0  EE                        out     dx, al                  ; set low byte
05B1  8A C4                     mov     al, ah
05B3  EE                        out     dx, al                  ; set hi byte
05B4  FB                        sti

05B5  C3                        ret
05B6                    Br9     endp

05B6                    Br115   proc NEAR

05B6  FA                        cli
05B7  BA 1506                   mov     dx, PIT03               ; Baud gen. control port
05BA  B0 B6                     mov     al, 0B6h                ; cntr2, lsb+msb, mode 3
05BC  EE                        out     dx, al
05BD  BA 1504                   mov     dx, PIT02               ; cntr2 data
05C0  B8 0007                   mov     ax, BndRD115            ; 115kHz Baud rate
05C3  EE                        out     dx, al                  ; set low byte
05C4  8A C4                     mov     al, ah
05C6  EE                        out     dx, al                  ; set hi byte
05C7  FB                        sti

```

```
05C8  C3                               ret
05C9                           Br115   endp

                               ;       -------------------
                               ;       Convert hex to binary
                               ;       -------------------
                               ;       input:
                               ;               ds:si --> character string
                               ;       output:
                               ;               ds:si --> terminating char
                               ;               ax = int, max 16 bits

05C9                           hex2bin proc    NEAR
05C9  51                               push    cx
05CA  52                               push    dx
05CB  B1 04                            mov     cl, 4
05CD  33 C0                            xor     ax, ax
05CF  8A 14                    h010:   mov     dl, [si]
05D1  80 CA 20                         or      dl, ' '
05D4  80 EA 30                         sub     dl, '0'
05D7  7C 19                            jl      h050
05D9  80 FA 09                         cmp     dl, 9
05DC  7E 0D                            jle     h030
05DE  80 EA 31                         sub     dl, 'a'-'0'
05E1  7C 0F                            jl      h050
05E3  80 FA 05                         cmp     dl, 5
05E6  7F 0A                            jg      h050
05E8  80 C2 0A                         add     dl, 10
05EB  D3 E0                    h030:   shl     ax, cl
05ED  0A C2                            or      al, dl
05EF  46                               inc     si
05F0  EB DD                            jmp     h010
05F2  5A                       h050:   pop     dx
05F3  59                               pop     cx
05F4  C3                               ret
05F5                           hex2bin endp

                               ;       -------------------
                               ;       Convert binary to hex
                               ;       -------------------
                               ;       input:
                               ;               ax = binary int
                               ;               ds:si --> 5-byte buffer (4 hex chars + null terminator)
                               ;       output:
                               ;               all regs except ax preserved

05F5  30 31 32 33 34 35 36+    hextab  db      '0123456789ABCDEF'
      37 38 39 41 42 43 44+
      45 46
0605                           bin2hex proc    NEAR
0605  53                               push    bx
0606  51                               push    cx
0607  52                               push    dx
0608  B9 0004                          mov     cx, 4
```

```
0608  BA 0F04               mov     dx, 0f04h
060E  03 F1                 add     si, cx
0610  32 FF                 xor     bh, bh
0612  88 3C                 mov     BYTE PTR [si], bh
0614  4E            bin2h1: dec     si
0615  8A D8                 mov     bl, al
0617  22 DE                 and     bl, dh
0619  2E: 8A 9F 05F5r       mov     bl, hextab[bx]
061E  88 1C                 mov     [si], bl
0620  86 D1                 xchg    dl, cl
0622  D3 E8                 shr     ax, cl
0624  86 D1                 xchg    dl, cl
0626  E2 EC                 loop    bin2h1
0628  5A                    pop     dx
0629  59                    pop     cx
062A  5B                    pop     bx
062B  C3                    ret
062C                bin2hex endp

                    ;==========================================================================
                    ; Hardware interrupt service routines
                    ;
                    ;       ---------------------------------
                    ;       Catch-all interrupt service routine
                    ;       ---------------------------------

062C                dummy_isr proc  FAR
062C  CF                    iret
062D                dummy_isr endp

                    ;       ----------------------
                    ;       Receiver ready interrupt
                    ;       ----------------------

062D                rxrdy_isr proc  FAR
062D  50                    push    ax                      ; save regs
062E  53                    push    bx
062F  52                    push    dx
0630  1E                    push    ds
0631  B8 0000s              mov     ax, DATA                ; point ds --> RAM data seg
0634  8E D8                 mov     ds, ax
0636  BA 1302               mov     dx, UART01              ; read status port
0639  EC                    in      al, dx
063A  8A E0                 mov     ah, al
063C  BA 1300               mov     dx, UART00
063F  EC                    in      al, dx                  ; read char, clear irq
0640  8B 1E 0400r           mov     bx, rx_inp              ; get rcv queue input pointer
0644  88 07                 mov     [bx], al                ; store char at ds:bx --> q-in
0646  FE C3                 inc     bl                      ; inc q-in pointer, 256 bytes long
0648  3B 1E 0402r           cmp     bx, rx_outp             ; check for q full
064C  74 09                 je      rxrd10                  ; jmp if so, don't update q-in pointer
064E  F6 C4 30              test    ah, 30h                 ; test for error bits
0651  75 04                 jnz     rxrd10                  ; jmp if any errors, don't update ptr
0653  89 1E 0400r           mov     rx_inp, bx
```

```
0657  B0 20            rxrd10: mov     al, EOI              ; send eoi to 8259
0659  BA 1100                  mov     dx, IC00
065C  EE                       out     dx, al
065D  1F                       pop     ds                   ; restore regs & return
065E  5A                       pop     dx
065F  5B                       pop     bx
0660  58                       pop     ax
0661  CF                       iret
0662                   rxrdy_isr   endp


                       ;       -------------------------
                       ;       Transmitter ready interrupt
                       ;       -------------------------

0662                   txrdy_isr proc  NEAR
0662  50                       push    ax
0663  53                       push    bx
0664  52                       push    dx
0665  1E                       push    ds
0666  BA 1302                  mov     dx, UART01
0669  EC                       in      al, dx
066A  A8 01                    test    al, 1
066C  74 1B                    jz      txrd04
066E  B8 0000s                 mov     ax, DATA             ; ds --> RAM
0671  8E D8                    mov     ds, ax
0673  BA 1300                  mov     dx, UART00
0676  8B 1E 0406r              mov     bx, tx_outp          ; ds:bx --> transmit output queue
067A  3B 1E 0404r              cmp     bx, tx_inp           ; queue empty?
067E  74 09                    je      txrd04               ; y -- exit isr
0680  8A 07                    mov     al, [bx]             ; pick up next char to xmit
0682  EE                       out     dx, al               ; send it
0683  FE C3                    inc     bl                   ; increment q ptr (q size = 256 chars)
0685  89 1E 0406r              mov     tx_outp, bx
0689  BA 1100          txrd04: mov     dx, IC00             ; send eoi to 8259
068C  B0 20                    mov     al, EOI
068E  EE                       out     dx, al
068F  1F                       pop     ds
0690  5A                       pop     dx
0691  5B                       pop     bx
0692  58                       pop     ax
0693  CF                       iret
0694                   txrdy_isr endp

                       ;       -------------------------
                       ;       Transmitter ready interrupt soft
                       ;       -------------------------

0694                   txrdy_s_isr proc  NEAR
0694  50                       push    ax
0695  53                       push    bx
0696  52                       push    dx
0697  1E                       push    ds
0698  BA 1302                  mov     dx, UART01
```

```
069B  EC                      in      al, dx
069C  A8 01                   test    al, 1
069E  74 1B                   jz      txrds4
06A0  B8 0000s                mov     ax, DATA            ; ds --> RAM
06A3  8E D8                   mov     ds, ax
06A5  BA 1300                 mov     dx, UART00
06A8  8B 1E 0406r             mov     bx, tx_outp         ; ds:bx --> transmit output queue
06AC  3B 1E 0404r             cmp     bx, tx_inp          ; queue empty?
06B0  74 09                   je      txrds4              ; y -- exit isr
06B2  8A 07                   mov     al, [bx]            ; pick up next char to xmit
06B4  EE                      out     dx, al              ; send it
06B5  FE C3                   inc     bl                  ; increment q ptr (q size = 256 chars)
06B7  89 1E 0406r             mov     tx_outp, bx
                              ;mov     dx, IC00            ; send eoi to 8259
                              ;mov     al, EOI
                              ;out     dx, al
06BB  1F              txrds4: pop     ds
06BC  5A                      pop     dx
06BD  5B                      pop     bx
06BE  58                      pop     ax
06BF  CF                      iret
06C0                  txrdy_s_isr endp

                      ;       ----------------------------
                      ;       Time base interrupt - 10.2 Hz
                      ;       ----------------------------

06C0                  timebase_isr proc NEAR
06C0  50                      push    ax
06C1  52                      push    dx
06C2  BA 1600                 mov     dx, WATCHD          ; reset watchdog timer
06C5  EE                      out     dx, al
06C6  BA 1100                 mov     dx, IC00
06C9  B0 20                   mov     al, EOI
06CB  EE                      out     dx, al
06CC  5A                      pop     dx
06CD  58                      pop     ax
06CE  CF                      iret
06CF                  timebase_isr endp


                      ;       ---------------------------
                      ;       Programmable timer interrupt
                      ;       ---------------------------

06CF                  timer_isr proc  NEAR
06CF  50                      push    ax
06D0  52                      push    dx
06D1  BA 1100                 mov     dx, IC00
06D4  B0 20                   mov     al, EOI
06D6  EE                      out     dx, al
06D7  5A                      pop     dx
06D8  58                      pop     ax
06D9  CF                      iret
```

```
     06DA                    timer_isr endp

                             ;       -------------------------
                             ;       Waveform generator interrupt
                             ;       -------------------------

     06DA                    wgen_isr proc  NEAR
     06DA  50                        push   ax
     06DB  52                        push   dx
     06DC  BA 1100                   mov    dx, IC00
     06DF  B0 20                     mov    al, EOI
     06E1  EE                        out    dx, al
     06E2  5A                        pop    dx
     06E3  58                        pop    ax
     06E4  CF                        iret
     06E5                    wgen_isr endp

                             ;       ------------------------------
                             ;       Waveform generator timer interrupt
                             ;       ------------------------------

     06E5                    wgtim_isr proc  NEAR
     06E5  50                        push   ax
     06E6  52                        push   dx
     06E7  BA 1100                   mov    dx, IC00
     06EA  B0 20                     mov    al, EOI
     06EC  EE                        out    dx, al
     06ED  5A                        pop    dx
     06EE  58                        pop    ax
     06EF  CF                        iret
     06F0                    wgtim_isr endp

                             ;=======================================================================
                             ; Software interrupt service routines
                             ;
                             ;       ----------------------------
                             ;       Get single char from serial port
                             ;       ----------------------------
                             ;
                             ;       Returns: ax = FFFF if no char present
                             ;                ax = ascii code if char present

     06F0                    getchar proc    FAR
     06F0  53                        push   bx
     06F1  1E                        push   ds
     06F2  B8 0000s                  mov    ax, DATA
     06F5  8E D8                     mov    ds, ax
     06F7  8B 1E 0402r               mov    bx, rx_outp          ; now ds:bx --> rcv queue
     06FB  B8 FFFF                   mov    ax, -1               ; set up for empty return
     06FE  3B 1E 0400r               cmp    bx, rx_inp           ; queue empty?
     0702  74 0A                     je     getc01               ; y -- return
     0704  8A 07                     mov    al, [bx]             ; get char from queue
     0706  FE C3                     inc    bl                   ; point to next q entry
     0708  89 1E 0402r               mov    rx_outp, bx
```

```
070C  32 E4                    xor     ah, ah                  ; zero high byte of char
070E  0B C0            getc01: or      ax, ax                  ; set condition code
0710  1F                       pop     ds
0711  5B                       pop     bx
0712  CF                       iret
0713                   getchar endp


                       ;       ------------------------------------
                       ;       Output single character to serial port
                       ;       ------------------------------------
                       ;       call with:
                       ;               al = char to output
                       ;               ah == 0 --> wait for queue space if queue is full
                       ;               ah != 0 --> return immediately if queue is full
                       ;       returns
                       ;               ah = 0  if char was placed in queue
                       ;               ah != 0 if char was not placed in queue
                       ;
                               assume  ds:DATA, es:nothing
0713                   putchar proc    FAR
0713  FB                       sti
0714  53                       push    bx
0715  52                       push    dx
0716  1E                       push    ds
0717  8B D0                    mov     dx, ax                  ; copy char to send
0719  B8 0000s                 mov     ax, DATA                ; address data segment
071C  8E D8                    mov     ds, ax
071E  8B 1E 0404r              mov     bx, tx_inp              ; get xmit queue input ptr
0722  8B C3                    mov     ax, bx                  ; copy it
0724  FE C0                    inc     al                      ; point to next location
0726  3B 06 0406r      putc03: cmp     ax, tx_outp             ; if inp == outp, queue is full
072A  75 08                    jne     putc03a                 ; jmp if not, ok to output char
072C  0A F6                    or      dh, dh                  ; is dh == 0?
072E  74 F6                    je      putc03                  ; y -- wait here for queue space
0730  8A E6                    mov     ah, dh                  ; n -- return with ah != 0
0732  EB 09                    jmp     SHORT putc04

0734  88 17            putc03a:mov     [bx], dl                ; store char in queue
0736  A3 0404r                 mov     tx_inp, ax              ; keep queue pointer
0739  CD 13                    int     TXRDY_S_INT             ; start transmission if necessary
073B  32 E4                    xor     ah, ah                  ; return with ah == 0
073D  8A C2            putc04: mov     al, dl
073F  1F                       pop     ds
0740  5A                       pop     dx
0741  5B                       pop     bx
0742  CF                       iret
0743                   putchar endp

                       ;       ------------------
                       ;       EEPROM Write Routine
                       ;       ------------------
                       ;       Input:
                       ;               ds:si  buffer containing data
```

```
                         ;              es:di  address to write
                         ;              cx     word count (address may not wrap over 64-word boundary)
                         ;       Output:
                         ;              ds:si  advanced by word count
                         ;              es:di  advanced by word count
                         ;              ax     return code
                         ;                       0, successful
                         ;                      -1, write didn't verify
                         ;                      -2, address conflict (protected ROM or 64-word violation)
                         ;                      -3, word count error (must be between 1 and 64)
                         ;              all other registers preserved
                         ;
                         ;       Note: this routine is copied to RAM and runs from there.
                         ;       This avoids code fetches from EEPROM while EEPROM write
                         ;       is being performed.

0743                     wree_rom proc   FAR
0743  51                         push    cx
0744  52                         push    dx

                         ;       First, validate address. Address must not overlap
                         ;       protected ROM segment, and must not cross 64-word boundary.

0745  B8 FFFD                    mov     ax, -3          ; error code ax == -3 if word count error
0748  E3 08                      jcxz    wree6           ; jmp if word count == 0
074A  83 F9 40                   cmp     cx, 64          ; check word count
074D  77 03                      ja      wree6           ; jmp if > 64
074F  EB 04 90                   jmp     wree7
0752  E9 0089            wree6:  jmp     wree2
0755  8C C0              wree7:  mov     ax, es          ; get target segment
0757  8B D7                      mov     dx, di          ; get target offset
0759  92                         xchg    ax, dx
075A  D1 E8                      shr     ax, 1           ; divide offset by 16
075C  D1 E8                      shr     ax, 1
075E  D1 E8                      shr     ax, 1
0760  D1 E8                      shr     ax, 1
0762  03 C2                      add     ax, dx          ; compute segment addr of start
0764  50                         push    ax              ; save it for later
0765  8C C0                      mov     ax, es
0767  8B D7                      mov     dx, di
0769  92                         xchg    ax, dx
076A  03 C1                      add     ax, cx          ; add word count
076C  03 C1                      add     ax, cx          ; ..make it byte count
076E  48                         dec     ax              ; subtract 1
076F  D1 E8                      shr     ax, 1
0771  D1 E8                      shr     ax, 1
0773  D1 E8                      shr     ax, 1
0775  D1 E8                      shr     ax, 1
0777  03 C2                      add     ax, dx          ; get segment address of end
0779  5A                         pop     dx
077A  3D FE00                    cmp     ax, PROT_ROM    ; result must not cross protected ROM
077D  73 2C                      jae     wre00           ; jmp if it does
077F  81 FA F000                 cmp     dx, Ghost1end
0783  73 0C                      jae     wree4
```

```
0785 81 FA E000              cmp    dx, Ghost1start
0789 72 06                   jb     wree4
078B B8 FFFE                 mov    ax, -2
078E EB 4E 90                jmp    wree2
0791 3D F000          wree4: cmp    ax, Ghost1end
0794 73 0B                   jae    wree5
0796 3D E000                 cmp    ax, Ghost1start
0799 72 06                   jb     wree5
079B B8 FFFE                 mov    ax, -2
079E EB 3E 90                jmp    wree2
07A1 25 FF80          wree5: and    ax, 0ff80h      ; now look at page address (upper 9 bits)
07A4 83 E2 80                and    dx, 0ff80h
07A7 3B C2                   cmp    ax, dx          ; must not cross 64-word boundary
07A9 74 10                   je     wree0           ; jmp if it's o.k.

07AB                  wre00:
07AB B8 FFFE                 mov    ax, -2          ; error code ax == -2 if address conflict
07AE 8C C2                   mov    dx, es
07B0 81 FA F000              cmp    dx, EEPROM1     ; exeption ONLY if es=EEPROM1
07B4 75 28                   jne    wree2           ; and di> KerAbv, NO exception if the same
07B6 83 FF F6                cmp    di, KerAbv      ; location adressed differently
07B9 72 23                   jb     wree2           ; last chance -- exception for above BOOT code

07BB                  wree0:
07BB BA 0200          wree3: mov    dx, WREE        ; enable eeprom access
07BE B0 0F                   mov    al, WREEEN
07C0 EE                      out    dx, al
                             ; rep    movsw          ; xfer data from RAM to eeprom in cx counts
07C1 A5                      movsw
07C2 E2 F7                   loop   wree3           ; rep cx time enable, write
07C4 32 C0                   xor    al, al
07C6 EE                      out    dx, al          ; disable eeprom access
07C7 B9 01F4                 mov    cx, 500         ; delay while write takes effect
07CA E2 FE                   loop   $
07CC 8B 44 FE                mov    ax, [si-2]
07CF B9 1450                 mov    cx, 5200        ; set timeout count (10 ms)
07D2 26: 3B 45 FE     wree1: cmp    ax, es:[di-2]   ; wait for write cycle to complete
07D6 E0 FA                   loopne wree1           ; check last word
07D8 B8 0000                 mov    ax, 0           ; error code ax == 0, doesn't change flags
07DB 74 01                   je     wree2           ; jmp if write was successful
07DD 48                      dec    ax              ; error code ax == -1 if write didn't verify

07DE 5A               wree2: pop    dx
07DF 59                      pop    cx
07E0 CF                      iret

07E1                  wree_rom endp
     = 009E           wree_len equ  $-wree_rom      ; number of bytes in this routine


                      ;==================================================================
                      ;
                      ;      ----------------------------
                      ;      Interrupt vector table
```

```
                               ;       (ROM, copied to RAM at startup)
                               ;       ----------------------------

     07E1  90                          even

                               ;       Hardware interrupts

     07E2  0000062Dsr        + romivt  dd      rxrdy_isr, txrdy_isr, wgen_isr, timebase_isr
           00000662sr        +
           000006DAsr        +
           000006C0sr
     07F2  000006CFsr        +         dd      timer_isr, wgtim_isr, dummy_isr, dummy_isr
           000006E5sr        +
           0000062Csr        +
           0000062Csr

                               ;       Software interrupts

     0802  000006F0sr        +         dd      getchar, putchar, wree_ram, txrdy_s_isr
           00000713sr        +
           00000700sr        +
           00000694sr
     0812  0000062Csr        +         dd      dummy_isr, dummy_isr, dummy_isr, dummy_isr
           0000062Csr        +
           0000062Csr        +
           0000062Csr

                               ;       ---------------------------
                               ;       Various code-segment constants
                               ;       ---------------------------

     0822  20 52 4F 4D 20 4D 6F+ idmsg   db      ' ROM Monitor program V  0.80', 0dh, 0ah, 0
           6E 69 74 6F 72 20 70+
           72 6F 67 72 61 6D 20+
           56 20 20 30 2E 38 30+
           0D 0A 00
     0841  62 79 3A 20 54 2E 20+ idmsg2  db      'by: T. Nolan/ M. Chmielowski', 0dh, 0ah, 0
           4E 6F 6C 61 6E 2F 20+
           4D 2E 20 43 68 6D 69+
           65 6C 6F 77 73 6B 69+
           0D 0A 00
     0860  4C 61 73 74 20 6D 6F+ idmsg3  db      'Last mod.  -  25th Nov. 1992', 0dh, 0ah, 0
           64 2E 20 20 2D 20 20+
           32 35 74 68 20 4E 6F+
           76 2E 20 31 39 39 32+
           0D 0A 00
     087F  0D 0A 3E 00           prompt  db      0dh, 0ah, '>', 0
     0883  0D 0A 00              crlf    db      0dh, 0ah, 0
     0886  57 72 69 74 65 20 45+ wrterr  db      'Write Err ', 0
           72 72 20 00
     0891  20 2D 2D 20 50 41 53+ pass    db      ' -- PASS', 0dh, 0ah, 0
           53 0D 0A 00
     089C  20 2D 2D 20 46 41 49+ fail    db      ' -- FAIL', 0dh, 0ah, 0
           4C 0D 0A 00
```

```
08A7  20 2A 44 45 4E 49 45+  denn    db      ' *DENIED', 0dh, 0ah, 0
      44 0D 0A 00
08B2  52 41 4D 31 20 70 61+  bit0    db      'RAM1 pattern 55AA   ', 0
      74 74 65 72 6E 20 35+
      35 41 41 20 20 20 00
08C7  52 41 4D 31 20 70 61+  bit1    db      'RAM1 pattern AA55   ', 0
      74 74 65 72 6E 20 41+
      41 35 35 20 20 20 00
08DC  52 41 4D 31 20 62 79+  bit2    db      'RAM1 byte access    ', 0
      74 65 20 61 63 63 65+
      73 73 20 20 20 20 00
08F1  52 41 4D 31 20 61 64+  bit3    db      'RAM1 address        ', 0
      64 72 65 73 73 20 20+
      20 20 20 20 20 20 00
0906  49 6E 74 65 72 72 75+  bit4    db      'Interrupt controller', 0
      70 74 20 63 6F 6E 74+
      72 6F 6C 6C 65 72 00
091B  38 32 35 35 20 50 50+  bit5    db      '8255 PPI chip       ', 0
      49 20 63 68 69 70 20+
      20 20 20 20 20 20 00
0930  38 32 35 34 20 74 69+  bit6    db      '8254 timer chip     ', 0
      6D 65 72 20 63 68 69+
      70 20 20 20 20 20 00
0945  55 41 52 54 20 63 68+  bit7    db      'UART chip           ', 0
      69 70 20 20 20 20 20+
      20 20 20 20 20 20 00
095A  45 45 50 52 4F 4D 31+  bit8    db      'EEPROM1 access      ', 0
      20 61 63 63 65 73 73+
      20 20 20 20 20 20 00
096F  44 53 52 20 61 63 74+  bit9    db      'DSR active          ', 0
      69 76 65 20 20 20 20+
      20 20 20 20 20 20 00
0984  52 41 4D 32 20 70 61+  bit10   db      'RAM2 pattern 55AA   ', 0
      74 74 65 72 6E 20 35+
      35 41 41 20 20 20 00
0999  52 41 4D 32 20 70 61+  bit11   db      'RAM2 pattern AA55   ', 0
      74 74 65 72 6E 20 41+
      41 35 35 20 20 20 00
09AE  52 41 4D 32 20 62 79+  bit12   db      'RAM2 byte access    ', 0
      74 65 20 61 63 63 65+
      73 73 20 20 20 20 00
09C3  52 41 4D 32 20 61 64+  bit13   db      'RAM2 address        ', 0
      64 72 65 73 73 20 20+
      20 20 20 20 20 20 00
09D8  45 45 50 52 4F 4D 32+  bit14   db      'EEPROM2 access      ', 0
      20 61 63 63 65 73 73+
      20 20 20 20 20 20 00
09ED  45 45 50 52 4F 4D 20+  bit15   db      'EEPROM KERNEL access', 0
      4B 45 52 4E 45 4C 20+
      61 63 63 65 73 73 00
                                     even
0A02  08B2r 08C7r 08DCr   +  bitmsg  dw      bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8, bit9, bit10, bit11,
      08F1r 0906r 091Br   +  bit12, bit13, bit14, bit15
      0930r 0945r 095Ar   +
```

```
     096Fr 0984r 0999r   +
     09AEr 09C3r 09D8r   +
     09EDr

                                       even
0A22 63 63 64 20 20 20 20+             db     ??filename        ; 8 byte file name
     20
0A2A 31 36 3A 34 33 3A 32+             db     ??time            ; 8 byte time
     35
0A32 31 31 2F 32 35 2F 39+             db     ??date            ; 8 byte date
     32

0A3A                           CODE    ends
                                       end    start
```

Symbol Name	Type	Value
??DATE	Text	"11/25/92"
??FILENAME	Text	"ccd "
??TIME	Text	"16:43:25"
??VERSION	Number	0202
@CPU	Text	0101H
@CURSEG	Text	CODE
@FILENAME	Text	CCD
@WORDSIZE	Text	2
ADCP	Number	1400
BIN2H1	Near	CODE:0614
BIN2HEX	Near	CODE:0605
BIT0	Byte	CODE:08B2
BIT1	Byte	CODE:08C7
BIT10	Byte	CODE:0984
BIT11	Byte	CODE:0999
BIT12	Byte	CODE:09AE
BIT13	Byte	CODE:09C3
BIT14	Byte	CODE:09D8
BIT15	Byte	CODE:09ED
BIT2	Byte	CODE:08DC
BIT3	Byte	CODE:08F1
BIT4	Byte	CODE:0906
BIT5	Byte	CODE:091B
BIT6	Byte	CODE:0930
BIT7	Byte	CODE:0945
BIT8	Byte	CODE:095A
BIT9	Byte	CODE:096F
BITMSG	Word	CODE:0A02
BNDRD115	Number	0007
BNDRD2	Number	0027
BNDRD4	Number	0014
BNDRD8	Number	000A
BNDRD9	Number	004E
BR115	Near	CODE:05B6
BR2	Near	CODE:057D
BR40	Near	CODE:056A
BR8	Near	CODE:0590
BR9	Near	CODE:05A3
BYTESTA	Near	CODE:0097
BYTESTB	Near	CODE:00B7
CRLF	Byte	CODE:0883
DENN	Byte	CODE:08A7
DIAG00	Near	CODE:02DA
DIAG01	Near	CODE:0313
DIAG02	Near	CODE:0319
DIAG03	Near	CODE:02F1
DIAG04	Near	CODE:02FE
DIAGF	Near	CODE:030B
DMP001	Near	CODE:04AF
DMP010	Near	CODE:04CE
DUMMY_ISR	Far	CODE:062C

DUMPMEM	Near	CODE:04AD
EEPN1	Number	0008
EEPN2	Number	0002
EEPN3	Number	0001
EEPROM1	Number	F000
EEPROM2	Number	D000
EE_BUF	Word	DATA:045C
EOI	Number	0020
FAIL	Byte	CODE:089C
GETC01	Near	CODE:070E
GETCHAR	Far	CODE:06F0
GETCH_INT	Number	0010
GETM01	Near	CODE:034D
GETM02	Near	CODE:035E
GETM04	Near	CODE:037A
GETM05	Near	CODE:037D
GETM10	Near	CODE:037F
GETMSG	Near	CODE:0338
GHOST1END	Number	F000
GHOST1START	Number	E000
GOUSER	Near	CODE:042D
GO_USERADD	Far	CODE:04AA
H010	Near	CODE:05CF
H030	Near	CODE:05EB
H050	Near	CODE:05F2
HEX2BIN	Near	CODE:05C9
HEXTAB	Byte	CODE:05F5
IC00	Number	1100
IC01	Number	1102
IDMSG	Byte	CODE:0822
IDMSG2	Byte	CODE:0841
IDMSG3	Byte	CODE:0860
KERABV	Number	FFF6
MONITR	Near	CODE:02BB
MSGBUF	Byte	DATA:040A
MSG001	Near	CODE:03A5
MSG002	Near	CODE:03AF
MSG004	Near	CODE:03B8
MSG006	Near	CODE:03BF
PASS	Byte	CODE:0891
PIT00	Number	1500
PIT01	Number	1502
PIT02	Number	1504
PIT03	Number	1506
PPI00	Number	1200
PPI01	Number	1202
PPI02	Number	1204
PPI03	Number	1206
PR040	Near	CODE:03D1
PR042	Near	CODE:03D9
PR044	Near	CODE:03E1
PR046	Near	CODE:03E9
PR048	Near	CODE:03F1
PR050	Near	CODE:03F9

PR052	Near	CODE:0400
PR054	Near	CODE:0409
PR060	Near	CODE:0412
PR065	Near	CODE:0423
PR066	Near	CODE:0444
PR070	Near	CODE:0450
PR072	Near	CODE:0464
PR074	Near	CODE:046A
PR080	Near	CODE:0471
PR082	Near	CODE:0484
PR084	Near	CODE:0489
PR099	Near	CODE:048D
PRL10	Near	CODE:0506
PRL12	Near	CODE:050E
PRL14	Near	CODE:051F
PRL15	Near	CODE:0521
PRL16	Near	CODE:0529
PRL17	Near	CODE:0537
PRL99	Near	CODE:055A
PROCESS	Near	CODE:03C4
PROGLOAD	Near	CODE:04F8
PROMPT	Byte	CODE:087F
PROT_ROM	Number	FE00
PUTC03	Near	CODE:0726
PUTC03A	Near	CODE:0734
PUTC04	Near	CODE:073D
PUTCHAR	Far	CODE:0713
PUTCH_INT	Number	0011
PUTMSG	Near	CODE:0391
PWRUP_ERR	Word	DATA:0408
RAM1	Number	0000
RAM2	Number	2000
RAMIVT	Dword	DATA:0020
RAM_INTR	Dword	DATA:0000
ROMIVT	Dword	CODE:07E2
RXRD10	Near	CODE:0657
RXRDY_INT	Number	0008
RXRDY_ISR	Far	CODE:062D
RX_INP	Word	DATA:0400
RX_OUTP	Word	DATA:0402
RX_QUEUE	Byte	DATA:0600
STACK_L	Word	DATA:0A00
START	Far	CODE:0000
STOP	Number	1000
TDSTRT	Number	FE00
TEMP	Far	CODE:0569
TEST0A	Near	CODE:0012
TEST0B	Near	CODE:002D
TEST1A	Near	CODE:0049
TEST1B	Near	CODE:0066
TEST2A	Near	CODE:0084
TEST2AF	Near	CODE:00A3
TEST2B	Near	CODE:00A6
TEST2BF	Near	CODE:00C3

TEST3A	Near	CODE:00C7
TEST3B	Near	CODE:00D3
TEST3C	Near	CODE:00DF
TEST3D	Near	CODE:00ED
TEST3E	Near	CODE:00F9
TEST3F	Near	CODE:0105
TEST3G	Near	CODE:0114
TEST4A	Near	CODE:015D
TEST5	Near	CODE:016A
TEST6	Near	CODE:017A
TEST7	Near	CODE:01A8
TEST7A	Near	CODE:01C2
TEST7B	Near	CODE:01D4
TEST7C	Near	CODE:01EC
TEST7X	Near	CODE:01CA
TEST8	Near	CODE:0200
TEST8A	Near	CODE:0226
TEST8B	Near	CODE:0246
TEST8C	Near	CODE:0256
TEST8D	Near	CODE:0276
TIMEBASE_ISR	Near	CODE:06C0
TIMER_ISR	Near	CODE:06CF
TPTTRN	Number	55AA
TSTOFSE1	Number	0000
TSTOFSE2	Number	0000
TSTOFSE3	Number	FFF6
TXRD04	Near	CODE:0689
TXRDS4	Near	CODE:06BB
TXRDY_INT	Number	0009
TXRDY_ISR	Near	CODE:0662
TXRDY_S_INT	Number	0013
TXRDY_S_ISR	Near	CODE:0694
TX_INP	Word	DATA:0404
TX_OUTP	Word	DATA:0406
TX_QUEUE	Byte	DATA:0500
UART00	Number	1300
UART01	Number	1302
USRAOFS	Number	0FFC
USRASEG	Number	FD00
WAITEC	Near	CODE:02AE
WATCHD	Number	1600
WGEN	Number	1700
WGEN_ISR	Near	CODE:06DA
WGTIM_ISR	Near	CODE:06E5
WRE00	Near	CODE:07AB
WREE	Number	0200
WREE0	Near	CODE:07BB
WREE1	Near	CODE:07D2
WREE2	Near	CODE:07DE
WREE3	Near	CODE:07BB
WREE4	Near	CODE:0791
WREE5	Near	CODE:07A1
WREE6	Near	CODE:0752
WREE7	Near	CODE:0755

```
WREEEN                     Number 000F
WREE_LEN                   Number 009E
WREE_RAM                   Far    DATA:0700
WREE_ROM                   Far    CODE:0743
WRITE_USERADD              Near   CODE:0494
WRTEE_INT                  Number 0012
WRTERR                     Byte   CODE:0886
XDEB                       Near   CODE:055B
XXX                        Near   CODE:0320

Groups & Segments          Bit Size Align  Combine Class

CODE                       16  0A3A Para   Public  CODE
DATA                       16  0A00 Para   Public  DATA
```

```
/*----------------------------------------------------------------------
/* pload.c -- program loader for CCD board
 *
 * Syntax: pload binfile seg [port]
 *
 * T. Nolan 8/1/92
 */


#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <io.h>
char fname[67];
char cmd[20];
char buf[128];
char inbuf[128];
char *inbufp = inbuf;
char comstr[] = "com1";
int  comtbl[] = {0x3f8, 0x2f8};
int  port;
#define TXRDY 0x20
#define RXRDY 0x01

void getcom(char *, char, int);
void putcom(char *, int, int);
void comerr(char *);
void usage();


/* #pragma intrinsic(inp, outp) */

/*----------------------------------------------------------------------
void main(int ac, char **av)
{
    FILE *fp, *fpc;
    char *arg;
    int  seg;
    long len;
    long count;
    int  i;

    if(ac < 3) usage();

    for(i=1; i<ac; i++)
    {
        arg = av[i];

        switch(i)
        {
            case 1: strcpy(fname, arg);
                    fp = fopen(fname, "rb");
                    if(!fp)
                    {
                        perror(arg);
                        exit(4);
                    }
```

```
                    break;

            case 2: seg = strtoul(arg, &arg, 16);
                    break;


            case 3: if(*arg == '1' || *arg == '2')
                        comstr[3] = *arg;
                    else usage();
                    break;

            default: usage();
        }

    }

    len = filelength(fileno(fp));
    port = comtbl[comstr[3] - '1'];

    printf("CCD Binary Program Loader v1.00 - T. Nolan 8/1/92\n");
    printf("Loading file %s using %s\n", fname, comstr);


    for(count = 0; count < len; )
    {

        printf("Segment %04x\r", seg);
        i = len-count > 128 ? 128 : len-count;
        sprintf(cmd, "L%04x %x\r", seg, i);
        putcom(cmd, strlen(cmd), port);
        getcom(buf, '\n', port);
        if(*buf != 'L') comerr(buf);
        fread(buf, 1, i, fp);
        putcom(buf, i, port);
        getcom(buf, '>', port);
        if(*buf != '\r') comerr(buf);
        count += i;
        seg += (i >> 4);

    }

    printf("\n%ld bytes sent\n", len);
    exit(0);
}

/*---------------------------------------------------------------------------
void putcom(char *buf, int len, int port)
{
    while(len--)
    {
        while(!(inp(port+5) & TXRDY))
            if(inp(port+5) & RXRDY) *inbufp++ = inp(port);
        outp(port, *buf++);
    }
}
```

```
/*------------------------------------------------------------------------
void getcom(char *buf, char end, int port)
{
    int c;
    char *ip = inbuf;

    while(ip < inbufp)
    {
        if((*buf++ = *ip++) == end)
        {
            *buf = '\0';
            return;
        }
    }
    inbufp = inbuf;

    while(1)
    {
        while(!(inp(port+5) & RXRDY))
            if(kbhit())
            {
                *buf = '\0';
                return;
            }
        if((*buf++ = inp(port)) == end)
        {
            *buf = '\0';
            return;
        }
    }
}

/*------------------------------------------------------------------------
void comerr(char *buf)
{
    printf("\nComm error %s\n", buf);
    exit(1);
}

/*------------------------------------------------------------------------
void usage()
{
    printf("Usage: pload binfile seg [port]\n");
    exit(2);
}
```

```
Program StarTracker; {ST5.pas}

{ Compiler Options N- E- do NOT use cooprocessor iven if present }

{$N-,E-,R+,G-,D+,S+,V+,X+,L+,A+,I- }

{ Limit memory usage for remote system }

{$M$4000,$2000,$2000 }

{ REMOTE option is define in TPC.CFG to select remote compilation }

{ compilation Units to use }

uses
  StSTART,
  Stdef01,
  StAqs01, StAQ01, STMain01, StPpic01, StCpic04,
  {$IfNDef REMOTE } StDpic04, {$EndIf }
  StPCSG01, StTest, StSrch01, StRem1;

var
  i: byte;
  NofStEndS: byte;
  ch: char;
  IntMax: word;
  IntMaxI: byte;
  Strr: string;
  j: longInt;

var
  jj: char;

Begin

  { Power up test }

  {test;}
  {testV(1000000);}
  {testV(4);}
  testV(1);

  { initial search }

  repeat
    NofStTrack:=StSearch(StarSearch.StNTofind);
    until NofStTrack>0;

  { set initial tracking windows - larger then when locked on star }

  WinStX:=WinStXSrch;
  WinStY:=WinStYSrch;
  WinSepX:=WinSepXSrch;
  WinSepY:=WinSepYSrch;
  WinWBx:=WinWBxSrch;
  WinHBy:= WinStY+2*WinSepYSrch;
  IntWindowStars(Stars);
  {
  WinStX:=WinStXtr;
  WinStY:=WinStYtr;
```

```
WinSepX:=WinSepXtr;
WinSepY:=WinSepYtr;
WinWBx:=WinWBxtr;
WinHBy:= WinStY+2*WinSepYtr;
SlowDownX:= SlowDownXTr;
}
{ClrScr;}
MlevelTgl:=Mlevel;
FrameNoCounter:=0;

{ main tracking loop }

repeat
  begin

  { Check background and adjust exposure every FrameNoToCheckB }

  if Mlevel then MlevelTgl:=not MlevelTgl else MlevelTgl:=false;
  if ( (FrameNoCounter mod FrameNoToCheck=0) or
       (FrameNoCounter<FrameNoToCheckB) )
    then
      begin
      NofStStart:=0;
      NofStEnd:=NofStTrack+1;
      BcgrToAdj:=true;
      if (FrameNoCounter mod FrameNoToCheck=0) then DAC_LEDCheck:=true;
      end
    else
      begin
      if (FrameNoCounter mod FrameNoToBcgr=0) then BcgrToAdj:=true;
      if (StToDispl=0) or (StToDispl=9) or DAC_LEDtoADJ then
        begin
        NofStStart:=0;
        NofStEnd:=NofStTrack+1;
        end
      else
        begin
        NofStStart:=1;
        NofStEnd:=NofStTrack;
        end;
      end;

  { Set new tracking windows boarder if needed }

  if ( (Mlevel and not MlevelTgl) or (not Mlevel) ) then
    begin
    inc(AfterSrchCT);
    if BcgrToAdj or BcgrToAdjWas or DebugDsplModeWas then
      begin
      SetWindowBorder(Stars);
      end;
    SetWindowStars(StarS);
    end;

  { Exposure Image, Store tracking windows data }

  StoreStarsImageSt(ExposureTime);
  {GenSynch;}

  { Convert tracking window data to digital form }
```

```
for i:=NofStStart to NofStEnd do
  begin
  AccuireWpictureSt(StarS.StarRs[i]);
  end;

{ Decode scrambled data }

for i:=NofStStart to NofStEnd do
  begin
  {ClearImageAndStorage(1);}
  ProcessPictureSt(StarS.StarRs[i]);
  end;
{ClearImageAndStorage(1);}

{ Calculate subpixel positions of stars }

if ( (Mlevel and MlevelTgl) or (not Mlevel) ) then
  begin
  for i:=1 to NofStTrack do
    begin
    CalculateStar(StarS.StarRs[i]);
    {ClearImageAndStorage(1);}
    end;
  {ExpToAdjust:=true;}

  { Display tracking windows if in TEST mode }

  {$IfNDef REMOTE }
  if (StToDispl<>9) and not DAC_LEDCheck then
    begin
    case StToDispl of
      7: DisplayStar(StarS.StarRs[7]);
      8: begin
          IntMax:=0;
          for i:=1 to NofStTrack do
            begin
            if StarS.StarRs[i].Magn.StarIntMax>IntMAX then
              begin
              IntMax:=StarS.StarRs[i].Magn.StarIntMax;
              IntMaxI:=i;
              end;
            end;
           StToDispl:=IntMaxI;
         DisplayStar(StarS.StarRs[StToDispl]);
         end;
      else
        DisplayStar(StarS.StarRs[StToDispl]);
      end; {case}

    {ClearImageAndStorage(1);}
    end
  else
    begin
    for i:=NofStStart to NofStEnd do
      begin
      DisplayStar(StarS.StarRs[i]);
      if i=0 then
        begin
        {ClearImageAndStorage(5);}
```

```
       end;
      {ClearImageAndStorage(1);}
      end;
    end; {else}
  {$Else }
    {
    writeln( ' X ', StarS.StarRs[MainStar].Cntr.CntrX:3:2,
             ' Y ', StarS.StarRs[MainStar].Cntr.CntrY:3:2,
             ' m ', StarS.StarRs[MainStar].Magn. StarIntMax:4,
             ' M ', StarS.StarRs[MainStar].Magn.StMM:7:2,
             ' T ', ExposureTime:3,
             ' Cx', StarS.StarRs[MainStar].Wind.Xcenter:4:1,
             ' Cy', StarS.StarRs[MainStar].Wind.Ycenter:4:1);
    }
  {$EndIf }

  { Adjust exposure parameters, background and reposition windows }

  if BcgrToAdj then BcgrToAdjWas:=true else BcgrToAdjWas:=false;
  BcgrToAdj:=false;
  if ExpToAdjust then
    begin
    IntMax:=0;
    BcgrToAdj:=true;
    for i:=1 to NofStTrack do
      begin
      if StarS.StarRs[i].Magn.StarIntMax>IntMAX then
        begin
        IntMax:=StarS.StarRs[i].Magn.StarIntMax;
        IntMaxI:=i;
        end;
      end;
    MainStar:=IntMaxI;
    AdjustExposure(StarS.StarRs[MainStar]);
    {ClearImageAndStorage(1);}
    end
  else BcgrToAdj:=false;
  if PosToAdjust and (NofStTrack>0) then for i:=1 to NofStTrack do
    begin
    {BcgrToAdj:=true;}
    if true {PosNoAdjust and StarS.StarRs[i].St2N<>0} then
      begin
      RepositionWindow(StarS.StarRs[i]);
      {ClearImageAndStorage(1);}
      end;
    end;
  if (NofStStart=0) and (NofStEnd>NofStTrack) then
    begin
    if DAC_LEDtoAdj or DAC_LEDCheck then
      SetDACandLED(StarS.StarRs[0], StarS.StarRs[NofStEnd]);
    {ClearImageAndStorage(1);}
    end;

  { Count image frames }

  if FrameNoCounter<$FFFF then inc(FrameNoCounter)
                          else FrameNoCounter:=FrameNoToCheckB+1;
  end; {if}
 end;
until false;
```

End.

```
unit StSTART;

interface

implementation

{ copy program constant form EEPROM to RAM }


var
  FromSeg: byte absolute $D000:0000;
  ToSeg  : byte absolute $2000:0000;
const
  LenBytes = $D200;

BEGIN
  move(FromSeg, ToSeg, lenBytes);
END.
```

```
Unit STdef01; {STdef01.pas}

{ Definitions of global constant, data types, variables, records }

{ Procedure initializating dynamic variables                     }


Interface

{ uses REMOTE or TEST version I/O ports addresses }

{$IfNDef REMOTE }
uses
   CRT,DOS;
{$EndIf }


{ uses floating point aritmetic without cooprocessor }

Type
     float        = real;

{ CCD structure parameters }

const
     NofLsegments =    3;
     NofAlines    =  488;
     NofA2lines   =    NofAlines div 2; {244}
     NofPlines    =    2*NofAlines; {976}
     NofSlines    =    NofA2lines * NofLsegments; {732}
     NofDpixels   =   12;
     NofDLpixels  =   24;
     NofImPixLine = 1134;
     NofPixLine   =    NofImPixLine + NofDLpixels; {1158}
     NofSpixels   =    NofPixLine div NofLsegments; {386}
     NofSelements =    NofSpixels + NofDpixels; {398}
     NofDA3pixels =    NofSelements - NofDLpixels div NofLsegments; {390}
     NofEinLine   =    NofSelements * NofLsegments; {1194}
     CCDcenterXX  =  567;
     CCDcenterYY  =  490;
     {*********************}
     NofIMlinesMax=   NofAlines;
     NofPicLinesMax=  NofPlines;
     NofPointsMax =  104;        { NofPointsMax = 2*NofPoints + 8 }
     NofData      =    NofSelements;
     {*********************}
     NofImageMaxX = 1170;
     NofImageMinX = 36;
     NofImageMinY =  5;
     NofImageMaxY = 976;
     {*********************}

     { I/O Ports addresses }


     {$IfNDef REMOTE }
     Base       = $200;
     PIO_A0     = Base + $0;
     PIO_B0     = Base + $1;
     PIO_C0     = Base + $2;
```

```
PIO_WO     = Base + $3;

PIO_A1     = Base + $4;
PIO_B1     = Base + $5;
PIO_C1     = Base + $6;
PIO_W1     = Base + $7;

PIO_A2     = Base + $8;
PIO_B2     = Base + $9;
PIO_C2     = Base + $A;
PIO_W2     = Base + $B;

ADCouts    = PIO_A0;
{ADCouts is Word type Hi=PIO_B0 }
ADCstr     = PIO_C0;
SynchDr    = PIO_C0;


{
SDriver    = PIO_A2;
PDriver    = PIO_B2;
Integrator = SDriver;
}

{$Else }

PIO_A1     = $1200;
PIO_B1     = $1202;
PIO_C1     = $1204;
PIO_W1     = $1206;
ADCouts    = $1400;

{
Base       = $200;
PIO_A0     = Base + $0;
PIO_B0     = Base + $1;
PIO_C0     = Base + $2;
PIO_W0     = Base + $3;

PIO_A1     = Base + $4;
PIO_B1     = Base + $5;
PIO_C1     = Base + $6;
PIO_W1     = Base + $7;

PIO_A2     = Base + $8;
PIO_B2     = Base + $9;
PIO_C2     = Base + $A;
PIO_W2     = Base + $B;

ADCouts    = PIO_A0;
}
{ADCouts is Word type Hi=PIO_B0 }

ADCstr     = PIO_C1;
SynchDr    = PIO_C1;

{$EndIf }

AuxDr      = PIO_A1;
AnalDr     = PIO_B1;
```

```
   LedDr      = PIO_C1;

{ I/O Ports data bits functions }

{*************************************}
{ Signals from ADCouts }
   { 1-12 LSB-MSB of ADC }
   ADC_BUSY_0    = $1000;
   ADC_M_WAIT_0  = $2000;
   ADC_M_STOP    = $4000;
   ADC_Spare     = $8000;
{************************************}
  { Signals for ADCstr    }
   AD_CSr    = $40;{EXT_OUT1}
   ADCstrInit= $0;
{************************************}
  { Signals for AuxDr    }
    DACaddrOfs=     0;    { addr starts at $1 to $4 3 bits }
    MuxEn1    =    $8;
    Sclk      =    $10;
    SDin      =    $20;
    DtoA1_L   =    $40;
    Anal_EN   =    $80;
    Deselect = Anal_EN;
    AuxDrInit=   $0;
    DACAddrMask = $7;
    DACAddrEnMask = $F;
{************************************}
   { Signals for AnalDr  }
    MuxAddrOfs = $0;       { addr Mux 0 starts at $1 to $ 4 3 bits }
    MuxEn0      = $8;
    IAGI        = $10;
    ABGI        = $20;
    ABGC_En     = $40;
    Temp_En     = $80;
    AnalDrInit = $0;
    Mux0ChIntBuffOut   =  $3; { $5;was for old ver }
    Mux0ChHouse        =  $1;
    Mux0addrMask = $7;
    Mux0addrEnMask = $F;
{************************************}
   { Signals for LedDr  }
   LEDsOFF     = $F;      { LEDs at $1, $2, $4, $8 4 LEDs }
   LED1_OFF    = $1;
   LED2_OFF    = $2;
   LED3_OFF    = $4;
   LED4_OFF    = $8;
   MuxEn2      = $10;
   MuxEn3      = $20;
   ExtOut1     = $40;
   ExtOut2     = $80;
   {LedDrInit   = $0;}
   LedDrInit   = ExtOut2;
   Mux2_3EnMask= $30;
(*
 {**********************}
 { Signals for SDriver  }
   IntHold     =  0;
   ADstop      =  0;
   IntPolPos   =  1;
```

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

```
    IntSample   =  2;
    IntReset    =  4;
    AD_CS       =  8;

    SDG1_p      = 16;
    SDG2_p      = 32;
    SDG3_p      = 64;
    SDTR_p      =128;
    {****************}
    SDG_p       = SDG1_p or SDG2_p or SDG3_p;
    SDn         = ADstop;
    SDG1p       = SDG1_p or ADstop;
    SDG2p       = SDG2_p or ADstop;
    SDG3p       = SDG3_p or ADstop;
    SDGp        = SDG_p  or ADstop;
    SDTRp       = SDTR_p or ADstop;
    SDnR        = SDn    or IntReset;
    SDG1pR      = SDG1p  or IntReset;
    SDG2pR      = SDG2p  or IntReset;
    SDG3pR      = SDG3p  or IntReset;
    SDGpR       = SDGp   or IntReset;
    SDTRpR      = SDTRp  or IntReset;
    SDriverInit = $0;
  {*********************}
  { Signals for PDriver   }
    PDzero    =   0;
    PDIAGp    =   1;
    PDSRGs    =   2;
    PDSAG1p   =   4;
    PDIAGI    =   8;
    PDTMGp    =  16;
    PDABCp    =  32;
    PDSAG2p   =  64;
    PDABGI    = 128;
    PDn       = PDzero;
    PDnABm    = PDzero or PDABGI;
    PDMS1p    = PDTMGp or PDSAG1p;
    PDMS2p    = PDTMGp or PDSAG2p;
    PDMS1pABm = PDTMGp or PDSAG1p or PDABGI;
    PDMS2pABm = PDTMGp or PDSAG2p or PDABGI;
    PDIMp     = PDIAGp;
    PDIMpABm  = PDIAGp or PDABGI;
    PDMp      = PDTMGp;
    PDS1p     = PDSAG1p;
    PDS2p     = PDSAG2p;
    PDIMS1p   = PDIAGp or PDTMGp or PDSAG1p;
    PDIMS2p   = PDIAGp or PDTMGp or PDSAG2p;
    PDIMS1pABm= PDIAGp or PDTMGp or PDSAG1p or PDABGI;
    PDIMS2pABm= PDIAGp or PDTMGp or PDSAG2p or PDABGI;
    PDABMm    = PDIAGI or PDABGI;
    PDABn     = PDzero;
    PDABp     = PDABCp;
    PDABnM    = PDIAGI or PDABGI;
    PDABpM    = PDABp or PDIAGI or PDABGI;
    ABinnerL  = 285;
    PDriverInit= $0;
*)
  {********************************}
  { Signals for SynchDr   }
  {$IfNDef REMOTE }
```

```
   SynchB    = $1;
{$Else}
   SynchB    = $80;
{$EndIf}
   SynchZero = $0;

{ Initial parameters for exposure, LED, search, counters }
{ DACs, thresholds                                       }

{*********************}
   NofPoints    : word = 16;
   NofPointsInt : word = 4;
   AntiBl       : boolean = {false;} true;
   FrameNoCounter: word  = 0;
   FrameNoToCheckB: word = 3;
   FrameNoToBcgr: word = 10;
   FrameNoToCheck: word =50;
   ExposureTime : word = 50;
   ExposureTimeR: word = 50;
   ExposureTimeOld: word = 10;
   ExposureTimeMax: word = 1000;
   ExpTSrchStart  : word = 1;
   ExpTSrchMinO    : word = 10;
   ExpTSrchMaxO    : word = 100;
   Mlevel       : boolean = true; {false;}
   MlevelTgl    : boolean = true; {false;}
   ExpToAdjust  : boolean = true;
   ExpNoAdjust  : word    = 7;
   PosToAdjust  : boolean = true;
   DAC_LEDtoADJ : boolean = true;
   DAC_LEDCheck : boolean = false;
   DACtoAdj     : boolean = true;
   LEDtoAdj     : boolean = true;
   BcgrToAdj    : boolean = true;
   BcgrToAdjWas : boolean = false;
   DAC_Adj_Stb  : word    = 0;
   DAC_Adj_try  : word    = 0;
   LED_Adj_Stb  : word    = 0;
   LED_Adj_try  : word    = 0;
   LED_Line_Br  = 4;
   DAC_Line_Dr  = 6;
   DAC_LED_Width = 10;
   DAC_LED_Pos  = 150;
   SSwindow     : boolean = true;
   LedWindow    : boolean = true; {false;}
   LedTimeM     : word    = 100;
   LedTimeO     : word    =  20;
   LedTime1     : word    =  20;
   LedTime1Srch : word    =  10;
   LedTime1SrchSet: word  =  10;
   LedTimeMax   = 100;
   LedNumM      : word    = 4;
   LedNum       : word    = 4;
   DebugDsplMode: boolean = false; {true;}
   DebugDsplModeWas: boolean = false;
   {********************************}
   PicShifted   = true;
   PicNoShift   = false;
   Bits12       = 4096;
   Bit12        = 4095;
```

```
Bit12delta   =   16;
MaxLongInt   = 2147483647;
{***********************************}

SearchMax    : integer    = Bits12;
SearchMin    : integer    =     0;
LocSearchMax : integer    =     0;
LocSearchMin : integer    =     0;
SearchMaxX   : integer    =     0;
SearchMaxY   : integer    =     0;
DACOV        : word       =     0;
DAC1V        : word       = 1024;
DAC2V        : word       = 2524;
DAC3V        : word       = 1024;
DAC4V        : word       = 1024;
DAC5V        : word       = 1024;
DAC6V        : word       = 1024;
DAC7V        : word       = 1024;
Dac3VthStart : word       =  500;
Dac3VthSet   : word       =  500;
TempD        : word       =     0;

{ Star search initial parameters - # of star to search for }
{ initial windows settings, windows setting when locked    }

NofStMax          = 7;
NofStTrack        :word=2;
NofStStart        :word=0;
NofStEnd          :word=3;
NofDataSt         = 9; {9; to fit screen 80 colums}
NofIMLinesMaxSt   = 25; {13; to fit 80x25 mode} {25 to fit 80x50 VGA}
NofEinLineSt      = NofLsegments * NofDataSt; {27}
NofPlinesSt       = 2*NofIMLinesMaxSt; { 26 or 50 }
WinWmax           = (NofDataSt-1)*NofLsegments; {8*3=24 points *3=72 col}
WinHmax           = NofPlinesSt-2;  {24 or 48}
{ Window is WinHmax+1 x WinWmax+1 points }
WinStXtr  = 11;
WinStYtr  = 11;
WinStXSrch= 17;
WinStYSrch= 17;
WinStX     : byte = 17; {5;}
WinStY     : byte = 17; {5;}
WinSepXSrch= 0;
WinSepYSrch= 0;
WinSepXtr = 3;
WinSepYtr = 3;
WinSepX    : byte = 0;  {3;}
WinSepY    : byte = 0;
WinHBy     : byte = WinStYSrch+2*WinSepYSrch; {5;}
WinWBxSrch= 2;
WinWBxtr  = 2;
WinWBx     : byte = WinWBxSrch;
WinOfsX      : word = 1; {5;}
WinOfsY      : word = 15; {5;}
SlowDownXSrch  = 2; {2;}
SlowDownYSrch  = 0;
SlowDownXtr    = 1; {2;}
SlowDownYtr    = 0;
SlowDownX    : byte     = SlowDownXSrch; {2;}
SlowDownY    : byte     = SlowDownYSrch;
```

```
    Mscale          = 20;
    Mscale0         = 1; {0.9;}
    Mscale1         = 1; {1.1;}
    Xrepos          = 0.51;
    Yrepos          = 0.51;
    AfterSrchSW     : word = 5;
    AfterSrchCT     : word = 0;

    MainStar        : byte = 1;
    StToDispl       : byte = 8;

    TestMode        : boolean=false; {true;}
    SearchXON       : boolean=false; {true;}
    SearchYON       : boolean=false; {true;}

    LineToShowMax   : word = 10;
    LineToShowMin   : word = 1;

    StarMove        : boolean=false;
    StpAtN          = 20; {10;}
    StpAtpointsMax  = NofLsegments*StpAtN;
    {StpAtpointsMax  = NofSelements;} {398}
    StpAtlinesMax   = 80; {40;}

    { search area, hot spot rejection criteria }

    VoltMax         = 24;
    PixNotLook      : word = 50;
    LineNotLook     : word = 20;
    LineWreq        : word = 1;
    LineWreqmax     : word = 30;
    LineGapRequired : word = 10;
    PixWreq         : word = 1;
    PixWreqmax      : word = 17;



    { data type structure definitions for image data handling, }
    { search data storing, tracking windows and parameters,    }
    { housekiping data                                         }

Type
    TVoltages       = array[0..VoltMax] of word;
    TstpAtpoint     = array[0..StpAtpointsMax+1] of word;
    TstpAtpoints    = array[0..StpAtlinesMax]  of TstpAtpoint;
    TMBank          = ( Bank_A, Bank_B);
    TDataMout       = array[1..NofPointsMax] of byte;
    Tshift          = ( PicSh0, PicSh1 );
    TshiftFloat     = array[Tshift] of float;
    TshiftInteger   = array[Tshift] of integer;

    TDataMinSt      = array[1..NofDataSt] of word;
    TImageLineSt    = array[1..NofLsegments] of TDataMinSt;
    TImageLinesSt   = array[1..NofIMLinesMaxSt] of TImageLineSt;
    TStImageSt      = TImageLinesSt;

    TPictureLineSt  = array[1..NofEinLineSt] of word;
    TPictureLinesSt = array[1..NofPlinesSt] of TPictureLineSt;
    TStPictureSt    = TPictureLinesSt;
```

```
TStAqPar=record
  NofAQPdump   : word;
  NofAQpix     : word;
  NofAQLines   : word;
  NofAQLdump   : word;
  NofNegLines  : word;
  end; {record}

TStWind=record
  WinTT      : integer;
  WinBB      : integer;
  WinLL      : integer;
  WinRR      : integer;
  Xcenter    : float;
  Ycenter    : float;
  XcenterP   : float;
  YcenterP   : float;
  WinChanged : boolean;
  end; {record}

TStMagn=record
  ExpT         : word;
  IntT         : word;
  StMM         : float;
  StarIntMax   : word;
  end; {record}

TStCntr=record
  CntrX        : float;
  CntrY        : float;
  end; {record}

TStBckgr=record
  BgL0, BgR0, BgT0, BgB0 : float;
  BgL1, BgR1, BgT1, BgB1 : float;
  BGrad0X, BGrad0Y, BGrad1X, BGrad1Y: float;
  Bg0, Bg1, Bg: float
  end; {record}

TStAux=record
  Xslw, XstLB, XendLB, XstRB, XendRB, Xsep, YstB, YendB: integer;
  XstSt, XendSt, Yslw, YstSt, YendSt: integer;
  end; {record}

TStTest=record
  Xtest: word;
  Ytest: word;
  Mtest: word;
  end; {record}


TStarR=record
  StN          : word;
  St2N         : word;
  StPicture    : TStPictureSt;
  StImage      : TStImageSt;
  StAqPar      : TStAqPar;
  Wind         : TStWind;
  Magn         : TStMagn;
  Cntr         : TStCntr;
```

```
      Bckgr         : TStBckgr;
      StAux         : TStAux;
      StTest        : TStTest;
      end; {record}

    TStarsG=record
      NofAQPdumpG  : word;
      NofAQpixG    : word;
      NofAQLinesG  : word;
      NofAQLdumpG  : word;
      NofStoLinesG : word;
      BB_DAC, BB_DAC0, BB_DAC6,
      BB_LED0w0, BB_LED1w0, BB_LED0w6, BB_LED1w6,
      BB_LED0, BB_LED1, BB_LED: float;
      end; {record}

    TStarRs=array[0..NofStMax] of TStarR;

    TStarS=record
      StarsG: TStarsG;
      StarRs: TStarRs;
      end; {record}

    TstSearchD  =record
      LineGapB : word;
      LineB    : word;
      LineT    : word;
      LineM    : word;
      LineW    : word;
      LineGapT : word;
      PixGapL  : word;
      PixL     : word;
      PixM     : word;
      PixR     : word;
      PixW     : word;
      PixGapR  : word;
      ExpF     : word;
      ExpO     : word;
      end;

    TstSearchDA = array[0..NofStMax] of TstSearchD;

    TStarSearchR=record
      StNfound     : word;
      StNtracked   : word;
      StNsearch    : word;
      StExpMaxSrch: word;
      DAC3VTh      : word;
      StNTofind    : word;
      BadLines     : word;
      StSearchDA   : TstSearchDA;
      end;

{ $IfNDef REMTST}

    { variables storing housekiping data, search data, image data }
    { and tracked star parameters                                 }

var
    {DataSD1            : TDataMout;}
```

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

```
    {PDataSD1          : Pointer;}
    StarS             : TStarS;
    StpAtpoints       : TstpAtpoints;
    Voltages          : TVoltages;
    StarSearch        : TStarSearchR;
{ $EndIF}

{ procedure for data initialization and DELAY procedure for remote system }

procedure Delay(Dms:integer);


Implementation


{ $IfNDef REMTST}
function Power2(n:word):word;
  var
    i:word;
    P:longInt;
  begin
  if i=0 then Power2:=1
  else
    begin
    P:=1;
    for i:=1 to n do P:=2*P;
    end;
  Power2:=P;
  end;

procedure SetWindOfs;
  var
    ofs, i: integer;
  begin
  {
  Ofs:= WinWmax div 2 - WinStX div 2 - WinSepX - WinWBx - SlowdownX + 1;
  if Ofs>=0 then WinOfsX:=Ofs else WinOfsX:=0;
  Ofs:= WinHmax div 2 - WinStY div 2 - WinSepY          - SlowdownY;
  if Ofs>=0 then WinOfsY:=Ofs else WinOfsY:=0;
  }
  for i:=0 to NofStMax do
    begin
    StarS.StarRs[i].StN:=i;
    StarS.StarRs[i].St2N:=Power2(i);
    StarS.StarRs[i].Wind.WinChanged:=true;
    end;
  StarS.StarsG.BB_DAC:=0;
  StarS.StarsG.BB_LED:=0;
  end;

{ $EndIF}

{ $IfDef REMOTE }
procedure Delay(Dms:integer);
  var
    j,i, Del1ms, DelNms:integer;
  begin
  DelNms:=52;
  Del1ms:=52;
  case Dms of
```

```
    0:     ;
    1: begin
         for j:=0 to Del1ms do
           begin
         end;
       end;
    else
      begin
      for i:=0 to Dms do
        begin
        for j:=0 to DelNms do
          begin
          end;
        end;
      end; {else}
   end; {case}
  end;
{ $Else}
{
procedure Delay(Dms:integer);
  begin
  CRT.delay(Dms);
  end;
}
{ $EndIf }

Begin
  { $IfNDef REMTST}
  {PDataSD1:=@DataSD1;}

  SetWindOfs;
  StarSearch.StNTofind:=1;

  { $EndIf}
End.
```

```
Unit STaqs01; {StaqS01.pas}

interface

{ Initial I/O ports setting }

  {
  procedure ByPassOFF;
  procedure ByPassON;
  }
  procedure SetPIO(ByPass:boolean);

implementation

uses
  STdef01, StPCSG01, StAQ01;
(*
procedure FillDataMSD(var Data: TDataMout);
  var
    i, j: word;
    OK: boolean;
  begin
  { OUT 1 }
  OK := false;
  repeat
    begin
    Nofpoints := 2+NofPointsInt+NofPointsInt+2;
    if NofPoints>NofPointsMax then
      begin
      dec(NofPointsInt);
      end
    else
      begin
      OK := true;
      end; {else}
    end; {rep}
  until OK;
  j := 1;
  Data[j] := SDG3pR;
  Inc(j);
  Data[j] := SDG3pR;
  Inc(j);
  for i := 1 to NofPointsInt do
    begin
    { Dual slope Integrator }
    Data[j] := (IntSample) + SDG1p;
    Inc(j);
    end;
  for i := 1 to NofPointsInt do
    begin
    Data[j] := (IntSample+IntPolPos) + SDG2p;
    Inc(j);
    end; {for}
  Data[j] := ADstop + IntHold + AD_CS + SDG2p;
  Inc(j);
  Data[j] := IntHold + AD_CS + SDG2p;
  end;
*)

procedure SetPIO(ByPass:boolean);
```

```
  begin
  {$IfNDef REMOTE }
  Port[PIO_W0]:=$92;  { Ports A & B inpts, C out mode 0 }
  {To be removed }
  Port[PIO_W2]:=$92;  { Ports A & B inpts, C out mode 0 }
  {$EndIf }
  Port[PIO_W1]:=$80;  { Ports A, B & C outs     mode 0 }
  (*
  if ByPass then
    begin
    PCSG_ToByPass;
    Port[PIO_W2]:=$80;  { Ports A, B & C outs     mode 0 }
    end
  else
    begin
    Port[PIO_W2]:=$92;  { Ports A & B inpts, C out mode 0 }
    end;
  *)
  end;
(*
procedure ByPassOFF;
  begin
  Port[PIO_W2]:=$92;  { Ports A & B inpts, C out mode 0 }
  end;

procedure ByPassON;
  begin
  PCSG_ToByPass;
  Port[PIO_W2]:=$80;  { Ports A & B outs, C out mode 0 }
  end;
*)

procedure SetPIOouts;
  begin
  {$IfNDef REMOTE }
  Port[PIO_A0]:=0;
  Port[PIO_B0]:=0;
  Port[PIO_C0]:=0;
  Port[PIO_A1]:=0;
  Port[PIO_B1]:=0;
  Port[PIO_C1]:=0;
  Port[PIO_A2]:=0;
  Port[PIO_B2]:=0;
  Port[PIO_C2]:=0;
  {$Else }
  Port[PIO_A1]:=0;
  Port[PIO_B1]:=0;
  Port[PIO_C1]:=0;
  {$EndIf }
  end;


Begin
  {$IfNDef REMTST }
  SetPIOouts;
  SetPIO(True);
  LedOff;
  {FillDataMSD(DataSD1);}
  {$EndIf }
End.
```

```
Unit STaq01; {STaq01.pas}

{ procedure controlling CCD,  data transfer, and data digitalization }

{ procedures for software emulation of existing hardware - for       }
{ performance comparision                                             }

Interface

uses
  STdef01;

procedure GenSynch;      { Synchronization with osciloscope for testing }

{ LED low level }
procedure LedOn(N:word);
procedure LedOff;
{$IfNDef REMOTE }
{ For5 PC only - memory refresh control }
procedure ONmemRefresh;
procedure OFFmemRefresh;
{$EndIf }

{ Software emulator }


(*
procedure ABClockASM(MM: word; Ml: byte);
procedure TransfT;
procedure Transf0;
 procedure Transf1;
procedure Transf2;
procedure Transf3;
procedure Shift1_3inA;
procedure Shift1_3inB;
procedure ShiftLines1_3ASM(NL: word; MB: byte);
procedure StoreIMlinesASM(NL: word; MB: byte);
procedure SDclearASM(NT: word);
procedure Get1_3IMlineASMSt(var DataMin  : TDataMinSt; WinW: word );
procedure ClearImageAndStorage(Times: word);
*)

{ A to D conversion control }
procedure AccuireWpictureSt(var Star: TStarR);
{ calculating CCD clocking to get selected windows }
procedure SetAccuisitionParSt(var Star: TStarR);
{ Search control }
function  FastSearchAQ:word;
{ Low level I/O }
function  AnalDrData:byte;
{ Housekiping data accuisition }
procedure SetMuxOCh(n:byte);
procedure SetMuxOIntBuffOut;
{ Antibluming clocking control }
procedure SetABGI_ON;
procedure SetABGI_OFF;
{ CCD interlacing control }
procedure SetIAGI_ON;
procedure SetIAGI_OFF;
{
```

```
function  ADmux1(del:word):word;
}

Implementation

uses
  StAux01, StPCSG01, StAqs01;

procedure GenSynch;
  begin
  Port[SynchDr]:=Port[SynchDr] or SynchB;
  Port[SynchDr]:=Port[SynchDr] and not SynchB;
  end;

procedure LedOn(N:word);
  var
    nn:byte;
  begin
  case N of
    0  : nn:=$0;
    1  : nn:=$1;
    2  : nn:=$1+$2;
    3  : nn:=$1+$2+$4;
    4  : nn:=$1+$2+$4+$8;
    else
      nn:=$0;
    end; {case}
  Port[LedDr]:=( Port[LedDr] or LEDsOFF ) and not nn;
  end;

procedure LedOff;
  begin
  Port[LedDr]:=Port[LedDr] or LEDsOFF;
  end;

function  AnalDrData:byte;
  begin
  AnalDrData:=Port[AnalDr];
  end;

procedure SetABGI_ON;
  begin
  Port[AnalDr]:=Port[AnalDr] or ABGI;
  end;

procedure SetABGI_OFF;
  begin
  Port[AnalDr]:=Port[AnalDr] and not ABGI;
  end;

procedure SetIAGI_ON;
  begin
  Port[AnalDr]:=Port[AnalDr] or IAGI;
  end;

procedure SetIAGI_OFF;
  begin
  Port[AnalDr]:=Port[AnalDr] and not IAGI;
  end;
```

```
procedure SetMux0Ch(n:byte);
  begin
  Port[AnalDr]:=(Port[AnalDr] and not Mux0addrMask) or MuxEn0 or (n and Mux0addrMask);
  end;

procedure SetMux0IntBuffOut;
  begin
  Port[AnalDr]:=(Port[AnalDr] and not Mux0addrMask) or MuxEn0 or Mux0ChIntBuffOut;
  end;

{$IfNDef REMOTE }
procedure OFFmemRefresh;
  begin
  ASM
    cli                 {3}
    mov     al, 54h
    out     43h, al
    mov     AL,   0
    OUT     41H, AL
    end;
  end;

procedure ONmemRefresh;
  begin
  ASM
    mov     al,54h
    out     43h,al
    mov     AL,18
    OUT     41H,AL
    sti
    end;
  end;

{$EndIf }


(*
  {$G+}
procedure ABClockASM(MM: word; Ml: byte);
  begin
    ASM
    cli
    push  bp
    mov   dx, PDriver
    mov   al, PDABp
    mov   ah, PDABn
    mov   cl, Ml              { Ml = PD1n , or Ml = PD1nM }
    mov   ch, cl
    or    ax, cx              { modify al and ah ffor Midle level }
    mov   bx, ABinnerL
    mov   cx, MM
    @2:
    mov   bp, cx          { save cx }
    mov   cx, bx          {cx:=ABinnerL}
    @1:
    out   dx, al          { PDABp }
    xchg  al, ah
    out   dx, al          { PDABn }
    xchg  al, ah
    loop  @1
```

```
    mov   cx, bp          { Restore cx }
    loop  @2
    mov   al, PDn
    out   dx, al          { AB ML }
    pop   bp
    sti
    end;
  end;
  {$G-}

  {$G+}
procedure TransfT;
  begin
  ASM
    cli                  {3}
    mov   dx, SDriver    {5}
    mov   al, SDTRpR     {5}
    mov   bl, SDnR       {5}
    out   dx, al         {3}     { SDTR P }
    xchg  al, bl         {3}
    out   dx, al         {3}     {all  N}
    sti                  {2}
    end;             {tot 40}
  end;
  {$G-}

  {$G+}
procedure Transf0;
  begin
  ASM
    cli                  {3}
    mov   dx, SDriver    {5}
    mov   al, SDGpR      {5}
    mov   bl, SDnR       {5}
    out   dx, al         {3}     { SDGp }
    xchg  al, bl         {3}
    out   dx, al         {3}     {all  N}
    sti                  {2}
    end;             {tot 40}
  end;
  {$G-}



  {$G+}
procedure Transf1;
  begin
  ASM
    cli                  {3}
    mov   dx, SDriver    {5}
    mov   al, SDGpR      {5}
    mov   ah, SDTRpR     {5}
    mov   bl, SDnR       {5}
    out   dx, al         {3}     {SDGp}
    xchg  al, ah         {3}
    out   dx, al         {3}     {SDTRp}
    xchg  al, ah         {3}
    out   dx, al         {3}     {SDGp}
    xchg  al, bl         {3}
    out   dx, al         {3}     {all  N}
```

```
    sti                   {2}
    end;              {tot 40}
  end;
  {$G-}


  {$G+}
procedure Transf2;
  begin
  ASM
    cli                   {3}
    mov   dx, SDriver     {5}
    mov   al, SDGpR       {5}
    mov   ah, SDTRpR      {5}
    mov   bl, SDnR        {5}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDTR P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDTR P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, bl          {3}
    out   dx, al          {3}       {all  N}
    sti                   {2}
    end;              {tot 52}
  end;
  {$G-}


  {$G+}
procedure Transf3;
  begin
  ASM
    cli                   {3}
    mov   dx, SDriver     {5}
    mov   al, SDGpR       {5}
    mov   ah, SDTRpR      {5}
    mov   bl, SDnR        {5}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDTR P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDTR P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDG_ P}
    xchg  al, ah          {3}
    out   dx, al          {3}       {SDTR P}
    xchg  al, bl          {3}
    out   dx, al          {3}       {all  N}
    sti                   {2}
    end;              {tot 64}
  end;
  {$G-}

  {$G+}
```

```
procedure Shift1_3inA;
  begin
  ASM
    cli                       {3}
    mov   dx, PDriver         {5}
    mov   al, PDSAG1p         {5}
    mov   ah, PDn             {5}
    out   dx, al              {3}         { S1 P }
    xchg  al, ah              {3}
    out   dx, al              {3}         { PD n }
    sti                       {2}
    end;                {tot 19}
  end;
  {$G-}

  {$G+}
procedure Shift1_3inB;
  begin
  ASM
    cli                       {3}
    mov   dx, PDriver         {5}
    mov   al, PDSAG2p         {5}
    mov   ah, PDn             {5}
    out   dx, al              {3}         { S2 P }
    xchg  al, ah              {3}
    out   dx, al              {3}         { PD n }
    sti                       {2}
    end;                {tot 19}
  end;
  {$G-}


  {$G+}
procedure StoreIMlinesASM(NL: word; MB: byte);
  begin
  if (NL=0) or TestMode then EXIT;
  ASM
    cli                       {3}
    mov     al, 54h
    out     43h, al
    mov     AL,   0
    OUT     41H, AL
    mov   cx, NL
    push  bp

    mov   ah, MB
    mov   al, SDGpR
    mov   si, ax

    mov   dx, SDriver
    mov   di, dx
    mov   dx, PDriver

    mov   bl, SDTRpR
    mov   bh, PDnABm          {5}

    mov   dx, SDriver
    mov   ax, si
    out   dx, ax
    mov   ax, bx
```

```
    out   dx, ax
    mov   al, SDnR
    out   dx, al
    mov   dx, PDriver

    @2:

    xchg  dx, di          {SDriver}
    mov   bp, cx          {Store cx}
    mov   cx, NofLSegments

    @1:
    mov   ax, si
    out   dx, ax
    mov   ax, bx
    out   dx, ax
    mov   al, SDnR
    out   dx, al
    loop  @1
    mov   cx, bp
    xchg  dx, di           {PDriver}
    mov   al, PDIMpABm

    out   dx, al
    mov   al, bh
    out   dx, al
    loop  @2
    mov   dx, SDriver
    mov   cx, NofLSegments+ 2
    @3:
    mov   ax, si
    out   dx, ax
    mov   ax, bx
    out   dx, ax
    mov   al, SDnR
    out   dx, al
    loop  @3
    mov   ah, PDn
    out   dx, ax
    pop   bp
    mov      al,54h
    out      43h,al
    mov      AL,18
    OUT      41H,AL
    sti
    end;
  end;
  {$G-}

  {$G+}
procedure ShiftLines1_3ASM(NL: word; MB: byte);
  begin
  if NL=0 then EXIT;
  ASM
    cli                   {3}
    mov     al, 54h
    out     43h, al
    mov     AL,   0
    OUT     41H, AL
```

```
    mov   ah, MB
    mov   al, SDTRpR
    mov   si, ax

    mov   al, SDnR
    mov   ah, PDn
    mov   di, ax

    mov   bl, SDGpR
    mov   bh, PDn

    mov   dx, SDriver
    mov   cx, NL

    @1:
    mov   ax, si
    out   dx, ax
    mov   ax, bx
    out   dx, ax
    mov   ax, di
    out   dx, al
    loop  @1

    mov      al,54h
    out      43h,al
    mov      AL,18
    OUT      41H,AL
    sti
    end;
  end;
  {$G-}

  {$G+}
procedure SDclearASM(NT: word);
  begin
    ASM
    cli
    mov     al, 54h
    out     43h, al
    mov     AL,   0
    OUT     41H, AL
    mov    dx, SDriver
    mov    cx, NT
    mov    ah, SDG1pR
    mov    bl, SDG2pR
    mov    bh, SDG3pR
    @1:
    mov    al, bh
    out    dx, al          { SDG3 P, IntReset }
    mov    al, ah
    out    dx, al          { SDG1 P, IntReset }
    mov    al, bl
    out    dx, al          { SDG2 P, IntReset }
    loop   @1
    mov    al, SDnR
    out    dx, al          { SDn }
    mov      al,54h
    out      43h,al
    mov      AL,18
    OUT      41H,AL
```

```
    sti
    end;
  end;
  {$G-}

  {$G+}
procedure Get1_3IMlineASMSt(var DataMin  : TDataMinSt; WinW: word );
  begin
  {GenSynch;}
    ASM
    cld
    cli
    mov     al, 54h
    out     43h, al
    mov     AL,   0
    OUT     41H, AL
    {mov     dx,  SDriver}
    mov     bx,  WinW
    les     di, DataMin
    lds     si, PDataSD1
    mov     ax,si
    push    bp
    mov     bp, NofPoints
    mov     si,ax
    mov     cx, bp
    mov     dx,  SDriver
    rep     outsb
    dec     bx
    cmp     bx, 0
    je      @3
    @1:
    mov     si,ax
    mov     cx, bp
    mov     dx,  SDriver
    rep     outsb
    mov     dx,  ADCouts
    insw
    dec      bx
    cmp      bx, 0
    jne      @1
    @3:

    mov     cx, 3
    @2:
    nop
    loop     @2

    mov     dx,  ADCouts
    insw
    mov      al, SDnR
    mov      dx, SDriver
    out      dx,al
    pop      bp
    mov      al,54h
    out      43h,al
    mov      AL,18
    OUT      41H,AL
    sti
    end;
  end;
```

```
{$G-}

procedure ClearImageAndStorage(Times: word);
  var ZZ, ii : word;
  begin
  if Times>0 then
    begin
    for ZZ := 1 to Times do
      begin
      {
      StoreIMlinesASM(NofA2lines+10, PDMS1pABm);
      StoreIMlinesASM(NofA2lines+10, PDMS2pABm);
      }
      {GenSynch;}
      StoreS1PCSG(NofA2lines+10);
      StoreS2PCSG(NofA2lines+10);
      end; {for}
    end;
  Transf3PCSG;
  {
  ByPassON;
  Transf3;
  }
  end;

procedure ClearImageAndStorageandSR(Times: word);
  var ZZ, ii : word;
  begin
  if Times>0 then
    begin
    for ZZ := 1 to Times do
      begin
      {
      StoreIMlinesASM(NofA2lines+10, PDMS1pABm);
      StoreIMlinesASM(NofA2lines+10, PDMS2pABm);
      }
      {GenSynch;}
      StoreS1PCSG(NofA2lines+10);
      StoreS2PCSG(NofA2lines+10);
      end; {for}
    end;
  Transf3PCSG;
  {
  ByPassON;
  Transf3;
  }
  SDclearPCSG(NofSelements+NofDLpixels);
  end;
*)

procedure AccuireWpictureSt(var Star: TStarR);
  var
    ZZ, ZZZ, i, Neg: word;
  begin
  {GenSynch;}
  Anal_ON;
  LoadDAC(DAC1V,1);
  with Star, Star.StAqPar do
    begin
    Neg:=NofLsegments*NofNegLines;
```

```
    if not testMode then
      begin
      {ShiftLines1_3ASM(NofLsegments*2, PDMS2p);}
      ShiftBlines1_3_PCSG(NofLsegments*2); { 2 empty lines due to 1+5 pulses }
      Transf3PCSG;
      {
      ByPassON;
      Transf3;
      }
      {GenSynch;}
      {SDclearASM(3*NofSelements);}
      SDclearPCSG(3*NofSelements+40);
      {ByPassON;}
      end;
    {GenSynch;}
    for ZZ := 1 to NofAQlines do
      begin
      for ZZZ := 1 to NofLsegments do
        begin
        if Neg=0 then
          begin
          {if Star.StN=0 then GenSynch;}
          {Transf1PCSG;}
          {Transf1;}
          {ByPassON;}
          {Shift1_3inB;}
          {Transf2PCSG;}
          {if Star.StN=3 then GenSynch;}
          Shift1_3inBtr2_PCSG;
          {Transf2;}
          end
        else dec(Neg);
        {GenSynch;}
        if NofAQPdump>0 then SDclearPCSG(NofAQPdump);
                             {SDclearASM(NofAQPdump);}
        {ByPassON;}
        {GenSynch;}
        {if Star.StN=0 then GenSynch;}
        {Get1_3IMlineASMSt( StImage[ZZ,ZZZ], NofAQpix );}
        {Get1_3IMlinePCSGSt( StImage[ZZ,ZZZ], NofAQpix );}
        GetPixADCwait_PCSG( StImage[ZZ,ZZZ], NofAQpix );
        Transf1PCSG;
        SDclearPCSG(NofDLpixels);
        {SDclearASM(NofSelements);}
        {ByPassON;}
        end; {for}
      end; {for}
    end; {with}
  {Anal_OFF;}
  end;

function  FastSearchAQ:word;
  var
    Z, ZZ, ZZZ, Nstp, NstpTotal, i, ii, j: word;
    xpoint, ypoint: word;
    StpAtp: TstpAtpoint;
    jinc: boolean;
    OnlyClear: boolean;
  begin
  {GenSynch;}
```

```
Anal_ON;
SetMuxOIntBuffOut;
{ByPassON;}
LoadDAC(DAC3V,3);
OnlyClear:=false;
{OnlyClear:=true;}
{GenSynch;}
{delay(5000);}
{SDclearASM(NofSelements*3);}
SDclearPCSG(NofSelements);
{ByPassON;}
  {ByPassON;}
for j:=0 to StpAtlinesMax do
  begin
  StpAtPoints[j, StpAtpointsMax+1]:=0;
  for i:=0 to 3*StpAtN do
    begin
    StpAtPoints[j,i]:=0;
    end;
  end;
j:=1;
NstpTotal :=  0;
for Z:=0 to 1 do
  begin
  for ZZ := 1 to NofA2lines do
    begin
    ii:=1;
    for ZZZ := 1 to NofLsegments do
      begin
      {GenSynch;}
      {Transf1PCSG;}
      {Transf1;}
      {ByPassON;}
      {if Z=0 then Shift1_3inA else Shift1_3inB;}
      {GenSynch;}
      if Z=0 then Shift1_3inAtr2_PCSG else Shift1_3inBtr2_PCSG;
      {GenSynch;}
      {Transf2PCSG;}
      {Transf2;}
      GenSynch;
      {SDclearASM(NofSelements);}
      if OnlyClear or ((Z=0) and (ZZ<=LineNotLook)) then
        begin
        SDclearPCSG(NofSelements);
        Nstp:=0;
        end
      else
        begin
        if PixNotLook>0 then
          begin
          SDclearPCSG(PixNotLook);
          Nstp:=SDclearPCSGstop(NofSelements-PixNotLook, StpAtN, StpAtp);
          end
        else Nstp:=SDclearPCSGstop(NofSelements, StpAtN, StpAtp);
        end;
      if not OnlyClear and (Nstp>0) then
        begin
        jinc:=true;
        ypoint:=2*(ZZ+Z*NofA2lines)-1;
        StpAtpoints[j,0]:=ypoint;
```

```
          NstpTotal:=NstpTotal+Nstp;
          StpAtpoints[j,StpAtpointsMax+1]:=StpAtpoints[j,StpAtpointsMax+1]+Nstp;
          if Nstp>StpAtN then
            begin
            Nstp:=StpAtN;
            StpAtpoints[j,StpAtpointsMax+1]:=999;
            end;
          if StpAtpoints[j,StpAtpointsMax+1]>999 then
             StpAtpoints[j,StpAtpointsMax+1]:=999;
          for i:=1 to Nstp do
            begin
            xpoint:=NofLsegments*(StpAtp[i]-1)+ZZZ;
            if ii<=3*stpAtN then StpAtpoints[j,ii]:=xpoint;
            inc(ii);
            end;
          end; {if}
        {ByPassON;}
        end; {for}
        if jinc then
          begin
          if (j<StpAtlinesMax) then
            begin
            inc(j);
            end
          else
            begin
            OnlyClear:=true;
            NstpTotal:=9999;
            end;
          jinc:=false;
          end; {if}
      end; {for}
    end; {for}
  StpAtPoints[0,0]:=j-1;
  FastSearchAQ:=NstpTotal;
  StpAtPoints[0,1]:=NstpTotal;
  Anal_OFF;
  end;

procedure SetAccuisitionParSt(var Star: TStarR);
  begin
  with Star.Wind, Star.StAqPar do
    begin
    if (WinBB>0) then
      begin
      NofAQLdump := word( (WinBB-1) div 2 );
      NofAQLines := word( (WinTT+1) div 2 - (WinBB-1) div 2 );
      NofNegLines:= 0;
      end
    else
      begin
      NofAQLdump := 0;
      NofAQLines := word( (WinTT+1) div 2 + (Abs(WinBB)+2) div 2 );
      NofNegLines:= (Abs(WinBB)+2) div 2;
      end;
    NofAQPdump := word( (WinLL-1) div 3 );
    NofAQpix   := word( (WinRR+2) div 3 - (WinLL-1) div 3 );
    end; {with}
  end;
```

```
(*

{G+}
function  ADmux1(del:word):word;
  const
    Rep=10;
  var
    i,j, AD:word;
  begin
  {
  Port[SDriver]:= (SDn or IntReset) and (not ADstop);
  i:=0;
  repeat
    AD:=PortW[SDriver] and (8192-1);
    inc(i);
  until (AD>4095) or (i>Rep);
  Port[SDriver]:= SDn or IntReset or ADstop;
  }
  Port[PDriver]:= PDn or PDmuxA0;
  for i:=0 to del do begin end;
  Port[SDriver]:= (SDn or IntReset) and (not ADstop);
  i:=0;
  repeat
    AD:=PortW[SDriver] and (8192-1);
    inc(i);
  until (AD>4095) or (i>Rep);
  Port[SDriver]:= SDn or IntReset or ADstop;
  Port[PDriver]:= PDn;
  ADmux1:=AD and 4095;
  end;
{G-}

*)


Begin
End.
```

```
Unit STMain01; {STMain01.pas}

{ CCD data decoding, tracking windows location }

Interface

uses
  StDef01;

procedure ProcessPictureSt(var Star: TStarR);
procedure SetWindowStars(var Stars: TStars);
procedure SetWindowBorder(var Stars: TStars);
procedure IntWindowStars(var Stars: TStars);

Implementation

uses
  STaq01, STPpic01, StCpic04;

{ Set TEST and Mapping window location }

procedure SetIntStar(var Star: TStarR);
  begin
  if Star.StN=0 then
    begin
    with Star.Wind, Star.StTest do
      begin
      if StarSearch.StNfound>=1 then WinLL:= DAC_LED_Pos
      else WinLL:=25;
      WinBB:=0-DAC_Line_Dr+1;
      WinRR:=WinLL+SlowDownXtr+DAC_LED_Width-1;
      WinTT:=WinBB+DAC_Line_Dr+4+LED_Line_Br;
      XcenterP:=round((WinLL+WinRR)/2);
      YcenterP:=round((WinBB+WinTT)/2);
      WinChanged:=true;
      end; {with}
    end
  else
    begin
    if StarSearch.StNfound>=Star.StN then
      begin
      with Star.Wind, Star.Cntr do
        begin
        XCenterP:=StarSearch.StSearchDA[Star.StN].pixM;
        YCenterP:=StarSearch.StSearchDA[Star.StN].LineM;
        WinChanged:=true;
        end; {with}
      end
    else
      begin
      with Star.Wind, Star.StTest do
        begin
        WinLL:=1156;{1168;}
        WinBB:=960;
        WinRR:=WinLL+WinWmax;
        WinTT:=WinBB+WinHmax;
        XcenterP:=round((WinLL+WinRR)/2);
        YcenterP:=round((WinBB+WinTT)/2);
        Xcenter:=XcenterP;
        YCenter:=YcenterP;
```

```
        WinChanged:=true;
        end; {with}
      end;
    end; {else}
  end;

procedure SetWindowStars(var Stars: TStars);
  var
    i: byte;
  begin
  for i:=1 to NofStEnd do
    begin
    if Stars.StarRs[i].Wind.WinChanged then
      begin
      SetWindows(Stars.StarRs[i]);
      SetAccuisitionParSt(Stars.StarRs[i]);
      Stars.StarRs[i].Wind.WinChanged:=false;
      end;
    end;
  end;

procedure SetWindowBorder(var Stars: TStars);
  var
    i: byte;
  begin
  if AfterSrchCT<=AfterSrchSW then
    begin
    WinStX:=WinStXSrch;
    WinStY:=WinStYSrch;
    WinSepX:=WinSepXSrch;
    WinSepY:=WinSepYSrch;
    WinWBx:=WinWBxSrch;
    WinHBy:= WinStY+2*WinSepYSrch;
    {IntWindowStars(Stars);}
    end
  else
    begin
    WinStX:=WinStXtr;
    WinStY:=WinStYtr;
    WinSepX:=WinSepXtr;
    WinSepY:=WinSepYtr;
    WinWBx:=WinWBxtr;
    WinHBy:= WinStY+2*WinSepYtr;
    SlowDownX:= SlowDownXTr;
    if BcgrToAdj then BcgrToAdjWas:=true else BcgrToAdjWas:=false;
    if DebugDsplMode then DebugDsplModeWas:=true else DebugDsplModeWas:=false;
    end;
  for i:=1 to NofStEnd do
    begin
    SetWindows(Stars.StarRs[i]);
    {
    SetAccuisitionParSt(Stars.StarRs[i]);
    Stars.StarRs[i].Wind.WinChanged:=false;
    }
    end;
  end;

procedure IntWindowStars(var Stars: TStars);
  var
    i: byte;
```

```
  begin
  for i:=0 to NofStTrack+1 do
    begin
    SetIntStar(Stars.StarRs[i]);
    SetWindows(Stars.StarRs[i]);
    SetAccuisitionParSt(Stars.StarRs[i]);
    end;
  if DebugDsplMode then DebugDsplModeWas:=false;
  end;

procedure ProcessPictureSt(var Star: TStarR);


  begin
  if not MlevelTgl then RepackPicSt(PicNoShift, Star)
                  else RepackPicSt(PicShifted, Star);
  end;

Begin
  {IntWindowStars(Stars);}
End.
```

```
unit STPpic01; {STPpic01.pas}

{ Exposure, image storing, search and image data decoding }

Interface

uses
  StDef01;

procedure StoreStarsImageSt(ExposureTime:word);
function  StoreFastSearch(ExposureTime:word):word;
procedure RepackPicSt( Lshift: boolean; var Star: TStarR);


Implementation

uses
  STaq01, StAQS01, StPCSG01;


function PutInADrange(AD:word):word;
  begin
  PutInADrange := AD {mod Bits12};
  end;


{ Control exposure time, LED On time, antibluming }
procedure Exposure( ExposureTime: word; Lshift: boolean );
  var
    LT: word;
  begin
  if Lshift then
    begin
    LT:=LedTime0;
    SetIAGI_ON;
    end
  else
    begin
    LT:=LedTime1;
    SetIAGI_OFF;
    end;
  if AntiBl then
    begin
    if ExposureTime <= LT then
      begin
      ABGClockPCSG(ExposureTime);
      end
    else
      begin
      if LT>0 then
        begin
        ABGClockPCSG(LT);
        end;
      LedOFF;
      ABGClockPCSG(ExposureTime-LT);
      end;
    end
  else
    begin
    if ExposureTime <= LT then delay(ExposureTime)
```

```
    else
      begin
      if LT>0 then delay(LT);
      LedOFF;
      if ExposureTime-LT>0 then delay(ExposureTime-LT);
      end;
    end; {else}
  SetIAGI_OFF;
  LedOFF;
  end;


{ Decode pixel data for single image line }
procedure RepackLineSt(Sl, Dl: word; var Star: TStarR);
  var
    ix, i, ixx, j: word;
  begin
  with Star, Star.StAqPar, Star.Wind do
    begin
    j := 1;
    for ix := 1 to NofAQpix do
      for i := 1 to NofLsegments do
        begin
        ixx:=3*NofAqPdump+3*(ix-1)+i;
        if (ixx>=WinLL) and (ixx<=WinRR) then
          begin
          StPicture[DL,j] := PutInADrange( StImage[SL,i,ix] );
          Inc(j);
          end; {if}
        end; {for}
      end; {with}
  end;


{ Decode image window data }
procedure RepackPicSt( Lshift: boolean; var Star: TStarR);
  var
    iy,j, i, il, ill: word;
    iyy, iyn: integer;
  begin
  j := 1;
  with Star.Wind do
    begin
    if Lshift then il := 1 else il := 0;
    if odd(WinBB) then
      begin
      if Lshift then ill := 1 else ill := 0;
      end
    else
      begin
      if Lshift then ill := 0 else ill := 1;
      end;
    end; {with}
  with Star.StAqPar, Star.Wind do
    begin
    for iy := 1 to NofAQLines do
      begin
      iyy := 2*(NofAQLdump) + 2*(iy-1) + il + 1;
      iyn := iyy - 2*NofNegLines;
      if (iyn>=WinBB) and (iyn<=WinTT) then
```

```
        begin
        RepackLineSt(iy, j+ill, Star );
        Inc(j,2);
        end
      else
        begin
        end;
      end; {for}
    end; {with}
  end;


{ Control Exposure and image transfer }
{ store tracking windows and dump rest of image }
procedure StoreStarsImageSt(ExposureTime:word);
  var
    i: byte;
  begin
  ClearIandS_PCSG(1);
  if not MlevelTgl then
    begin
    if LedWindow and (LedTime1>0) then LedON(LedNum);
    {GenSynch;}
    Exposure( ExposureTime, PicNoShift );
    end
  else
    begin
    if LedWindow and (LedTime0>0) then LedON(LedNum);
    Exposure( ExposureTime, PicShifted );
    end;
  with StarS.StarsG do
    begin
    NofAQLdumpG :=0;
    NofAQPdumpG :=0;
    NofStoLinesG:=0;
    {GenSynch;}
    for i:=NofStStart to NofStEnd do
      begin
      {
      ByPassON;
      SetABGI_ON;
      if i=0 then GenSynch;
      StoreIMLinesASM(Stars.StarRs[i].StAqPar.NofAQLdump-NofAQLdumpG, PDMS1pABm);
      StoreIMLinesASM(Stars.StarRs[i].StAqPar.NofAQLines-
      Stars.StarRs[i].StAqPar.NofNegLines, PDMS2pABm);
      SetABGI_OFF;
      }
      SetABGI_ON;
      {if i=0 then GenSynch;}
      StoreS1PCSG(Stars.StarRs[i].StAqPar.NofAQLdump-NofAQLdumpG);
      StoreS2PCSG(Stars.StarRs[i].StAqPar.NofAQLines-
      Stars.StarRs[i].StAqPar.NofNegLines);
      SetABGI_OFF;

      NofStoLinesG:=NofStoLinesG+(Stars.StarRs[i].StAqPar.NofAQLines-
                                  Stars.StarRs[i].StAqPar.NofNegLines)+2;
      NofAQLdumpG:=Stars.StarRs[i].StAqPar.NofAQLdump+
                   Stars.StarRs[i].StAqPar.NofAQLines-
                   Stars.StarRs[i].StAqPar.NofNegLines;
      end; {for}
```

```
    {GenSynch;}
    {
    ByPassON;
    ShiftLines1_3ASM(NofLsegments*(NofA2Lines-NofStoLinesG), PDMS2p);
    }
    ShiftBLines1_3_PCSG(NofLsegments*(NofA2Lines-NofStoLinesG) );
    end; {with}
  end;


{ Store image for scan }
function  StoreFastSearch(ExposureTime:word):word;
  var
    i: byte;
  begin
  {GenSynch;}
  ClearIandS_PCSG(2);
  {GenSynch;}
  LedTime1:=LedTime1Srch;
  if LedWindow and (LedTime1>0) then LedON(LedNum);
  Exposure( ExposureTime, {PicShifted} PicNoShift );
  {GenSynch;}
  ShiftALines1_3_PCSG(NofA2Lines+10);
  ShiftBLines1_3_PCSG(NofA2Lines+10);
  StoreS1PCSG(NofA2Lines);
  StoreS2PCSG(NofA2Lines);
  StoreFastSearch:=FastSearchAQ;
  end;

Begin
End.
```

```
Unit STCpic04;

{ Centroid calculation, window parameters adjusting }

Interface

uses
  STdef01;

const
  SSminDev=1e-10;

procedure CalculateStar(var Star: TStarR);
procedure AdjustExposure(var Star:TStarR);
procedure RepositionWindow(var Star: TStarR);
procedure SetWindows(var Star: TStarR);
procedure SetDACandLED(var Star0, Star6: TstarR);



Implementation


{ Calculate subregions inside tracking windows }
procedure SetWindows(var Star: TStarR);
  begin
  with Star.StAux, Star.Wind, Star.Bckgr do
    begin
    if (Star.StN>0) and (Star.StN<=NofStTrack) then
      begin
      XstSt  := round(XcenterP - WinStX div 2);
      YstSt  := round(YcenterP - WinStY div 2);
      XendSt := XstSt+WinStX-1;
      YendSt := YstSt+WinStY-1;
      Xcenter:= (XendSt+XstSt)/2;
      Ycenter:= (YendSt+YstSt)/2;
      Xslw   := XstSt-SlowDownX;
      Yslw   := YstSt-SlowDownY;
      XstLB:=0;
      XendLB:=0;
      XstRB:=0;
      XendRB:=0;
      if BcgrToAdj or DebugDsplMode then
        begin
        BgL0:=0;
        BgL1:=0;
        BgR0:=0;
        BgR1:=0;
        Star.Magn.StarIntMax:=0;
        XendLB := XstSt-WinSepX-1;
        XstLB  := XendLB-WinWBx+1;
        XstRB  := XendSt+WinSepX+1;
        XendRB := XstRB+WinWBx-1;
        YstB   := YstSt-WinSepY;
        YendB  := YendSt+WinSepY;
        Xslw   := XstLB-SlowDownX;
        Yslw   := YstB-SlowDownY;
        WinLL  := Xslw;
        WinRR  := XendRB;
        WinBB  := YstB;
```

```
      WinTT  := YendB;
      Xsep:=round(((XendRB+XstRB)/2+(XendLB+XstLB)/2)/2);
      if DebugDsplMode then
        begin
        WinLL := WinLL-WinOfsX;
        WinBB := WinBB-WinOfsY;
        WinRR := WinLL+WinWmax;
        WinTT := WinBB+WinHmax;
        end;
      end
    else
      begin
      WinLL  := XstSt-SlowDownX;
      WinBB  := YstSt-SlowDownY;
      WinRR  := XendSt;
      WinTT  := YendSt;
      end;

    end
  else
    begin
    XstLB:=0;
    XendLB:=0;
    XstSt:=0;
    XendSt:=0;
    XstRB:=0;
    XendRB:=0;
    end;
  end; {with}
end;


{ Calculate centroid }
procedure StarCentrCal(var Star: TStarR);

  { Calculate Background }
  procedure PicWinBackgr(var Star: TStarR);
    var
      ix, iy: word;
    begin
    with Star, Star.StAux, Star.Wind, Star.Bckgr do
      begin
      for ix:=XstLB to XendLB do
        begin
        for iy:=YstB to YendB do
          begin
          if odd(iy) then BgL1:=BgL1+StPicture[iy-WinBB+1,ix-WinLL+1] and $FFF
                     else BgL0:=BgL0+StPicture[iy-WinBB+1,ix-WinLL+1] and $FFF;
          end;
        end;
      for ix:=XstRB to XendRB do
        begin
        for iy:=YstB to YendB do
          begin
          if odd(iy) then BgR1:=BgR1+StPicture[iy-WinBB+1,ix-WinLL+1] and $FFF
                     else BgR0:=BgR0+StPicture[iy-WinBB+1,ix-WinLL+1] and $FFF;
          end;
        end;
      if not odd(WinHBy) then
        begin
```

```
    BgL0:=BgL0/(WinHBy div 2)/WinWBx;
    BgL1:=BgL1/(WinHBy div 2)/WinWBx;
    BgR0:=BgR0/(WinHBy div 2)/WinWBx;
    BgR1:=BgR1/(WinHBy div 2)/WinWBx;
    Bg0 :=(BgL0+BgR0)/2;
    Bg1 :=(BgL1+BgR1)/2;
    Bg  :=(Bg0+Bg1)/2;
    end
  else
    begin
    if not odd(YstB) then
      begin
      BgL0:=BgL0/(WinHBy div 2 + 1)/WinWBx;
      BgL1:=Bgl1/(WinHBy div 2)/WinWBx;
      BgR0:=BgR0/(WinHBy div 2 + 1)/WinWBx;
      BgR1:=BgR1/(WinHBy div 2)/WinWBx;
      Bg0 :=(BgL0+BgR0)/2;
      Bg1 :=(BgL1+BgR1)/2;
      Bg :=(Bg0*(WinHBy div 2 + 1) + Bg1*(WinHBy div 2) )/WinHBy;
      end
    else
      begin
      BgL0:=BgL0/(WinHBy div 2)/WinWBx;
      BgL1:=Bgl1/(WinHBy div 2 + 1)/WinWBx;
      BgR0:=BgR0/(WinHBy div 2)/WinWBx;
      BgR1:=BgR1/(WinHBy div 2 + 1)/WinWBx;
      Bg0 :=(BgL0+BgR0)/2;
      Bg1 :=(BgL1+BgR1)/2;
      Bg :=(Bg1*(WinHBy div 2 + 1) + Bg0*(WinHBy div 2) )/WinHBy;
      end;
    end;
  Bgrad0X:=(BgR0-BgL0)/Xsep;
  Bgrad1X:=(BgR1-BgL1)/Xsep;
  end; {with}
end;

{ Used for centroid calculation in Y direction }
function SumOverX(var Star: TStarR; iy:word): float;
  var
    ix: word;
    Xsum, BB, BBgr: float;
  begin
  Xsum:=0;
  with Star, Star.StAux, Star.Wind, Star.Bckgr do
    begin
    if not odd(iy) then
      begin
      BB:=Bg0;
      BBgr:=Bgrad0X;
      end
    else
      begin
      BB:=Bg1;
      BBgr:=Bgrad1X;
      end; {else}
    for ix:=XstSt to XendSt do
      begin
      Xsum:=Xsum+(StPicture[iy-WinBB+1, ix-WinLL+1]  and $FFF -
                                  (BB+(ix-Xcenter)*BBgr) );
      end; {for}
```

```
    SumOverX:=Xsum/(XendSt-XstSt+1); {WinStX;}
    end; {with}
  end;

{ Used for centroid calculation in X direction }
  function SumOverY(var Star: TStarR; ix:word): float;
  var
    iy: word;
    Ysum, BB, BB0, BB1: float;
    PP: word;
  begin
  Ysum:=0;
  with Star, Star.StAux, Star.Wind, Star.Bckgr do
    begin
    BB0:=Bg0+(ix-Xcenter)*Bgrad0X;
    BB1:=Bg1+(ix-Xcenter)*Bgrad1X;
    for iy:=YstSt to YendSt do
      begin
      if not odd(iy) then
        begin
        BB:=BB0;
        end
      else
        begin
        BB:=BB1;
        end; {else}
      PP:=StPicture[iy-WinBB+1, ix-WinLL+1] and $FFF;
      Ysum:=Ysum+(PP - BB);
      if PP>= Star.Magn.StarIntMax then Star.Magn.StarIntMax:=PP;
      end; {for}
    SumOverY:=Ysum/(YendSt-YstSt+1); {WinStY;}
    end; {with}
  end;

{ 2 dim centroid }
procedure StarCentr(var Star: TStarR);
  var
    i: word;
    Sum, Sxy: float;
  begin
  with Star.StAux, Star.Wind, Star.Cntr, Star.Magn do
    begin
    Sum  :=0;
    CntrX:=0;
    StarIntMax:=0;
    for i:=XstSt to XendSt do
      begin
      Sxy:=SumOverY(Star,i);
      Sum:=Sum+Sxy;
      CntrX:=CntrX+Sxy*i;
      end; {for}
    if Abs(Sum)>SSminDev then CntrX:=CntrX/Sum else CntrX:=0;{Xcenter;}
    Sum  :=0;
    CntrY:=0;
    for i:=YstSt to YendSt do
      begin
      Sxy:=SumOverX(Star,i);
      Sum:=Sum+Sxy;
      CntrY:=CntrY+Sxy*i;
      end; {for}
```

```
        if Abs(Sum)>SSminDev then CntrY:=CntrY/Sum else CntrY:=0;{Ycenter;}
        if not odd(WinStY) then
          begin
          StMM:=Sum/WinStY/ExposureTime*Mscale;
          end
        else
          begin
          if not odd(YstSt) then
            begin
            StMM:=Sum/WinStY/ExposureTime*Mscale*Mscale0;
            end
          else
            begin
            StMM:=Sum/WinStY/ExposureTime*Mscale*Mscale1;
            end; {else}
          end; {else}
        end; {with}
      end;

    {StarCentrCal}
    begin
    if BcgrToAdj then PicWinBackgr(Star);
    StarCentr(Star);
    end; {StarCentrCal}

  { Adjust window position if Star is not in center }
  procedure RepositionWindow(var Star: TStarR);
    begin
    with Star, Star.Wind, Star.Cntr, Star.StAux do
      begin
      WinChanged:=true;
      if (CntrX>=XstSt) and (CntrX<=XendSt) and
         (CntrY>=YstSt) and (CntrY<=YendSt) then
        begin
        {
        if abs(CntrX-XcenterP)>Xrepos then XCenterP:=CntrX;
        if abs(CntrY-YcenterP)>Yrepos then YCenterP:=CntrY;
        }
        XcenterP:=CntrX;
        YcenterP:=CntrY;
        end
      else
        begin
        WinChanged:=false;
        end;
      end; {with}
    end;

{ Adjust exposure if maximum is not inside limits }
procedure AdjustExposure(var Star:TStarR);
  begin
  if Star.Magn.StarIntMax < (7 * Bits12) div 10 then
    begin
    if ExposureTime < ExposureTimeMax then
      begin
      if ExposureTime<10 then ExposureTime:=ExposureTime*2+1
                         else ExposureTime := round(ExposureTime * 1.25)+1;
      if ExposureTime > ExposureTimeMax then ExposureTime := ExposureTimeMax;
      end;
    end;
```

```
  if Star.Magn.StarIntMax > (9 * Bits12) div 10 then
    begin
    if EXposureTime >= 1 then ExposureTime:=ExposureTime-1;
    if ExposureTime > 1 then
      begin
      ExposureTime := round(ExposureTime / 1.25);
      if ExposureTime < 1 then ExposureTime := 0;
      end;
    end;
  if ExpNoAdjust>0 then dec(ExpNoAdjust);
  if ExpNoAdjust=0 then ExpToAdjust:=false;
  end;

{ Control Exposure adjustment and switching main star }
procedure CalculateStar(var Star: TStarR);
  begin
  StarCentrCal(Star);
  if (Star.Bckgr.Bg<Bits12 div 2) and
     ( (Abs(Star.Cntr.CntrX-Star.Wind.Xcenter)>Xrepos) or
       (Abs(Star.Cntr.CntrY-Star.Wind.Ycenter)>Yrepos) )
    then
      begin
      PosToAdjust:=true;
      end;
  if (Star.StN=MainStar) and
     ( (Star.Magn.StarIntMax<(7*Bits12) div 10) or
       (Star.Magn.StarIntMax>(9*Bits12) div 10) )
     then
       begin
       ExpToAdjust:=true;
       end;
  end;


{ Control background usung LEDs }
procedure SetDACandLED(var Star0, Star6: TstarR);
  var
    ix,jy, j, j0, j1, dLed: integer;
    dDac: LongInt;
    DACok: boolean;
    LEDok: boolean;
  begin
  j:=0;
  DACok:=false;
  Stars.StarsG.BB_DAC:=0;
  for jy:=Star0.Wind.WinBB to 0 do
    for ix:=Star0.Wind.WinLL+SlowDownXtr to Star0.Wind.WinRR do
      begin
      Stars.StarsG.BB_DAC:=Stars.StarsG.BB_DAC+
        Star0.StPicture[jy-Star0.Wind.WinBB+1,ix-Star0.Wind.WinLL+1] and $FFF;
      inc(j);
      end;
  Stars.StarsG.BB_DAC:=Stars.StarsG.BB_DAC/j;

  dDac:=0;
  LEDok:=false;
  if Stars.StarsG.BB_DAC>20 then
    begin
    dDac:=3;
    DAC_Adj_Stb:=0;
```

```
  if Stars.StarsG.BB_DAC>500 then
    begin
    dDac:=round(Stars.StarsG.BB_DAC);
    end;
  if Stars.StarsG.BB_DAC>100 then dDac:= round(Stars.StarsG.BB_DAC) div 2;
  if Stars.StarsG.BB_DAC>50  then dDac:= round(Stars.StarsG.BB_DAC) div 3;
  if Stars.StarsG.BB_DAC>30  then dDac:= round(Stars.StarsG.BB_DAC) div 4;
  if (DAC1V+dDac<2048) then DAC1V:=DAC1V+dDac else DAC1V:=2048;
  end
else
  begin
  if Stars.StarsG.BB_DAC< 10 then
    begin
    DAC_Adj_Stb:=0;
    dDac:=-5;
    if Stars.StarsG.BB_DAC<  5 then dDac:=-10;
    if Stars.StarsG.BB_DAC<  2 then dDac:=-100;
    if (DAC1V>-dDAC) then DAC1V:=DAC1V+dDac else DAC1V:=0;
    end;
  end;
if (Stars.StarsG.BB_DAC<20) and (Stars.StarsG.BB_DAC>10) then
  begin
  inc(DAC_Adj_Stb);
  DACok:=true;
  end
else inc(DAC_Adj_try);
if DAC_Adj_try>5 then
  begin
  DACtoAdj:=false;
  DAC_LEDCheck:=false;
  end;
if DAC_Adj_Stb>5 then DACtoAdj:=false;
{DAC1V:=1024;}

j0:=0;
j1:=0;
Stars.StarsG.BB_LED0:=0;
Stars.StarsG.BB_LED1:=0;
for jy:=5 to Star0.Wind.WinTT do
  for ix:=Star0.Wind.WinLL+SlowDownXtr to Star0.Wind.WinRR do
    begin
    if odd(jy) then
      begin
      Stars.StarsG.BB_LED1:=Stars.StarsG.BB_LED1+
        Star0.StPicture[jy-Star0.Wind.WinBB+1,ix-Star0.Wind.WinLL+1] and $FFF;
      inc(j1);
      end
    else
      begin
      Stars.StarsG.BB_LED0:=Stars.StarsG.BB_LED0+
        Star0.StPicture[jy-Star0.Wind.WinBB+1,ix-Star0.Wind.WinLL+1] and $FFF;
      inc(j0);
      end;
    end;
Stars.StarsG.BB_LED0:=Stars.StarsG.BB_LED0/j0;
Stars.StarsG.BB_LED1:=Stars.StarsG.BB_LED1/j1;
Stars.StarsG.BB_LED:=(Stars.StarsG.BB_LED0+Stars.StarsG.BB_LED1)/2;

{if DACtoAdj then EXIT;}
dLed:=0;
```

```
if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC < 310 then
  begin
  LED_Adj_Stb:=0;
  if LedTime0<1 then LedTime0:=1;
  dLed:=round(LedTime0*0.10);
  if dLed<1 then dLed:=1;
  if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC < 250 then
    begin
    dLed:=round(LedTime0*0.25);
    if dLed<1 then dLed:=1;
    end;
  if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC < 140  then
    begin
    dLed:=round(LedTime0*0.50);
    if dLed<1 then dLed:=1;
    end;
  if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC < 100  then
    begin
    dLed:=round(LedTime0*2.00);
    if dLed<2 then dLed:=2;
    end;
  if LedTime0+dLed<LedTimeMax then LedTime0:=LedTime0+dLed
                              else LedTime0:=LedTimeMax;
  end;
if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC > 360 then
  begin
  LED_Adj_Stb:=0;
  if LedTime0<2 then begin LedTime0:=0; EXIT; end;
  dLed:=-1;
  if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC > 1000 then
    begin
    dLed:=-round(LedTime0*0.25);
    if dLed>-1 then dLed:=-1;
    end;
  if Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC > 500 then
    begin
    dLed:=-round(LedTime0*0.5);
    if dLed>-1 then dLed:=-1;
    end;
  if LedTime0>-dLed then LedTime0:=LedTime0+dLed else LedTime0:=0;
  end;
dLed:=0;
if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC < 310 then
  begin
  LED_Adj_Stb:=0;
  if LedTime1<1 then LedTime1:=1;
  dLed:=round(LedTime1*0.10);
  if dLed<1 then dLed:=1;
  if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC < 250 then
    begin
    dLed:=round(LedTime1*0.25);
    if dLed<1 then dLed:=1;
    end;
  if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC < 140  then
    begin
    dLed:=round(LedTime1*0.50);
    if dLed<1 then dLed:=1;
    end;
  if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC < 100  then
    begin
```

```
          dLed:=round(LedTime1*2.00);
          if dLed<2 then dLed:=2;
          end;
        if LedTime1+dLed<LedTimeMax then LedTime1:=LedTime1+dLed
                                    else LedTime1:=LedTimeMax;
        end;
      if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC > 360 then
        begin
        LED_Adj_Stb:=0;
        if LedTime1<2 then begin LedTime1:=0; EXIT; end;
        dLed:=-1;
        if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC > 1000 then
          begin
          dLed:=-round(LedTime1*0.25);
          if dLed>-1 then dLed:=-1;
          end;
        if Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC > 500 then
          begin
          dLed:=-round(LedTime1*0.5);
          if dLed>-1 then dLed:=-1;
          end;
        if LedTime1>-dLed then LedTime1:=LedTime1+dLed else LedTime1:=0;
        end;
    if (Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC<360) and
       (Stars.StarsG.BB_LED0-Stars.StarsG.BB_DAC>310) and
       (Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC<360) and
       (Stars.StarsG.BB_LED1-Stars.StarsG.BB_DAC>310) then
      begin
      inc(LED_Adj_Stb);
      LEDok:=true;
      end
    else inc(LED_Adj_Try);
    if LED_Adj_Stb>5 then LEDtoAdj:=false;
    if LED_Adj_try>5 then
      begin
      LEDtoAdj:=false;
      DAC_LEDCheck:=false;
      end;
   if DAC_LEDCheck and DACok and LEDok then DAC_LEDCheck:=false;
   if not LedtoAdj and not DACtoAdj then DAC_LEDtoAdj:=false;
   end;

Begin
End.
```

```
Unit STDpic04; {STDpic04.pas}

{ FOR TESTs ONLY - Display tracking windows }
{ NOT part of Star Tracker Software - No comments }

Interface

{$IfNDef REMOTE }

uses
  StDef01;

procedure DisplayStar(Star: TStarR);

{$EndIf }


Implementation

{$IfNDef REMOTE }

uses
  CRT, DOS, StAux01;

var
  h, m, s, hund : Word;

  X2s, Y2s, Xs2, Ys2, Ys,
  M2s, mm2s, Ms2, mms2, Ms, MMs: float;


const
  Lines80=true;
  PointsTest={true;} false;
  BusyADC: boolean = false;
  WaitADC: boolean = false;
  StopADC: boolean = false;
  Nsigm  : Integer = 0;

function Xsigmf(X, Y,M, mm: float; var Ys, Ms, mms: float): float;
  begin
  Xsigmf:=9999;
  case Nsigm of
     0: begin
        X2s:=0;
        Xs2:=0;
        Y2s:=0;
        Ys2:=0;
        Ms2:=0;
        m2s:=0;
        mm2s:=0;
        mms2:=0;
        inc(Nsigm);
        end;
    11: begin
        dec (Nsigm);
        Xsigmf:=Sqrt(  (X2s-Xs2*Xs2/Nsigm) )/(Nsigm-1);
        Ys     :=Sqrt( (Y2s-Ys2*Ys2/Nsigm) )/(Nsigm-1);
        Ms     :=Sqrt( (M2s-Ms2*Ms2/Nsigm) )/(Nsigm-1);
        mms     :=Sqrt( (mm2s-mms2*mms2/Nsigm) )/(Nsigm-1);
```

```
        X2s:=0;
        Xs2:=0;
        Y2s:=0;
        Ys2:=0;
        Ms2:=0;
        m2s:=0;
        mm2s:=0;
        mms2:=0;
        Nsigm:=0;
        end;
    else
      begin
      inc(Nsigm);
      X2s:=X2s+X*X;
      Xs2:=Xs2+X;
      Y2s:=Y2s+Y*Y;
      Ys2:=Ys2+Y;
      Ms2:=Ms2+M;
      M2s:=M2s+M*M;
      mms2:=mms2+mm;
      mm2s:=mm2s+mm*mm;
      end;
    end; {case}
  end;

function PutInRange(AD:word):word;
  begin
  if (AD and $1000) = $1000 then BusyADC:=false else BusyADC:=true;
  if (AD and $2000) = $2000 then WaitADC:=true else WaitADC:=false;
  if (AD and $4000) = $4000 then StopADC:=true else StopADC:=false;
  PutInRange := round((AD mod Bits12)/100);
  end;

procedure DisplayPic(Star: TStarR);
  var i,j: integer;
      xsigm: float;
  begin
  if StToDispl=7 then
    begin
    GetTime(h,m,s,hund);
    writeln( ' X ', StarS.StarRs[MainStar].Cntr.CntrX:3:2,
             ' Y ', StarS.StarRs[MainStar].Cntr.CntrY:3:2,
             ' m ', StarS.StarRs[MainStar].Magn. StarIntMax:4,
             ' M ', StarS.StarRs[MainStar].Magn.StMM:7:2,
             ' T ', ExposureTime:3,
             ' Cx', StarS.StarRs[MainStar].Wind.Xcenter:4:1,
             ' Cy', StarS.StarRs[MainStar].Wind.Ycenter:4:1,
             ' Time ',s:2,':',hund:2         );
    Xsigm:=Xsigmf(StarS.StarRs[MainStar].Cntr.CntrX,
                  StarS.StarRs[MainStar].Cntr.CntrY,
                  StarS.StarRs[MainStar].Magn.StMM,
                  StarS.StarRs[MainStar].Magn. StarIntMax,
                  Ys, Ms, mms );
    if Xsigm<9999 then writeln(' Xs ',Xsigm:5:3, ' Ys ', Ys:5:3,
                               ' X NEA ',Xsigm*30:5:3,'" Y NEA ',Ys*30:5:3,'"',
                               ' Ms ', Ms:4:2,' ms', mms:4:2);
    EXIT;
    end;
  ClrScr;
  with Star, Star.Wind, Star.Cntr, Star.StAux do
```

```
begin
for j:=WinTT downto WinBB do
  begin
  for i:=WinLL to WinRR do
    begin
    if not ( (Star.StN=0) or (Star.StN>NofStTrack) ) then
      begin
      if (j>=YstB ) and (j<=YendB ) and
         ((i>=Xslw) or (i>=XstLB)) and ((i<=XendRB) or (XendRB=0))
        then
          begin
          TextBackground(LightGray);
          TextColor(Black);
          end
        else
          begin
          TextBackground(Black);
          TextColor(White);
          end; {else}

      if (abs(i-CntrX)<0.501) and (abs(j-CntrY)<0.501)
        then TextBackground(Magenta);

      if (i>=XstSt) and (i<=XendSt) and
         (j>=YstSt) and (j<=YendSt) then
        begin
        if (abs(i-Xcenter)<0.501) and (abs(j-Ycenter)<0.501)
          then TextColor(LightRed)
          else TextColor(LightGreen);
        end
      else
        if (i>=Xslw) and ((i<=XstLB-1) or ((XstLB=0) and (i<XstSt)) ) and
           (j>=YstB ) and (j<=YendB )
          then TextColor(Brown)
        else
          if (j>=YstB)  and (j<=YendB) and
             ( ((i>=XstLB) and (i<=XendLB)) or
               ((i>=XstRB) and (i<=XendRB)) )
             then TextColor(Blue);
      end {if}
    else
      begin
      TextColor(White);
      TextBackground(Black);
      if (Star.Stn=0) then
        begin
        if (i<WinLL+SlowDownXtr) and ( (j<=0) or (J>=5) )
          then TextColor(Brown)
        else
          begin
          if (j<=0) then TextColor(Blue);
          if (J>=5) then TextColor(LightGreen);
          end; {if}
        end;
      end; {else}
    write(PutInRange(StPicture[(j-WinBB+1),(i-WinLL+1)] ):2);
    {write($FFF and StPicture[(j-WinBB+1),(i-WinLL+1)]:4);}
    if not BusyADC {and not WaitADC and not StopADC} then write(' ')
    else
      begin
```

```
        if     BusyADC {and not StopADC} then write('B');
        {
        if     BusyADC and     StopADC then write('X');
        if not BusyADC and     StopADC then write('S');
        }
        end;
      end; {for}
    if j<>WinBB then writeln;
    end; {for}
  if Lines80 then GoToXY(1,50) else GoToXY(1,25);
  TextBackground(LightGray);
  TextColor(Black);
  if not( (StN=0) or (StN=NofStTrack+1) ) then
    begin
    if Star.StN=MainStar then
      begin
      TextBackground(LightGray);
      TextColor(Blue);
      end;
    write('St ',StN:2);
    TextBackground(LightGray);
    TextColor(Black);
    write(' B ',Star.Bckgr.Bg:7:2,
        ' X ', Star.Cntr.CntrX:3:2,' Y ', Star.Cntr.CntrY:3:2,
        ' m ', Star.Magn. StarIntMax:4,' M ', Star.Magn.StMM:7:2,
        ' T ', ExposureTime:3

        ,' Cx', Star.Wind.Xcenter:4:1
        ,' Cy', Star.Wind.Ycenter:4:1

        );
    end
  else
  write('St ',StN:2,
        ' T ', ExposureTime:3,
        ' Bdac ',Stars.StarsG.BB_DAC:7:2,
        ' DAC ', DAC1V:4,
        ' Bled ', Stars.StarsG.BB_LED:4:1,
        ' TL0', LedTime0:3,
        ' TL1', LedTime1:3

        ,' Cx', Star.Wind.Xcenter:4:1
        ,' Cy', Star.Wind.Ycenter:4:1

        );
  NormVideo;
  end; {with}
end;

procedure DisplayImage(Star: TStarR);
 var i,j, k: word;
 begin
 ClrScr;
 with Star, Star.StAqPar do
   begin
   for j:= NofAqLines downto 1 do
     begin
     for i:=1 to NofAqPix do
       begin
       for k:=1 to 3 do
```

```
          begin
          if (i=NofAqPix) then
          write( PutInRange(StImage[j,k,i]):2)
          else
          write( PutInRange(StImage[j,k,i]):2,' ');
          end;
        end;
      writeln;
      end; {for}
    write('Star ',StN:2);
    end; {with}
  writeln;
  end;

procedure DisplayStar(Star: TStarR);
  var
    Ch:Char;
  begin
  DisplayPic(Star);
  {delay(2000);}
  {DisplayImage(Star);}
  {
  if  ((Mlevel and MlevelTgl) or (not Mlevel)) and
      ((StToDispl=9) or (Star.StN=StToDispl)) then DisplayPic(Star);
  }
  {
  if  ((Mlevel and MlevelTgl) or (not Mlevel)) and
      ((StToDispl=9) or (Star.StN=StToDispl)) then DisplayImage(Star);
  }
  if KeyPressed then
    begin
    Ch:=ReadKey;
    if Ch=#27 then halt;
    case Ch of
      '0': StToDispl:=0;
      '1': StToDispl:=1;
      '2': StToDispl:=2;
      '3': StToDispl:=3;
      '9': StToDispl:=9;
      '8': StToDispl:=8;
      '7': StToDispl:=7;
      ' ': StarMove := not StarMove;
      'D','d' : begin
                  DebugDsplMode:=not debugDsplMode;
                  DebugDsplModeWas:=true;
                end;
      'P': PCooler_ON;
      'p': PCooler_OFF;
      else begin end;
      end; {case}
    end; {if}
  end;

{$EndIf }

Begin
  {$IfNDef REMOTE }
  if Lines80 then TextMode(CO80+Font8x8);
  {$EndIf }
End.
```

```
unit StPCSG01;

{ Controls sequencer (PCSG) generating CCD clock pulses }

interface

uses
  StDef01;

  procedure PCSG_ToByPass; { for TEST - use software emulation }
  { Generate selected clock waveform several times  8 and 16 bit output}
  procedure ExecWF( WFnum, Times: word );
  procedure ExecWFV( WFnum, Times: word );
  { Wait for sequencer to stop }
  procedure Wait_PCSG_Finish;
  { Check if PCSG stoped }
  function  PCSG_Finish:boolean;
  { Get number of generated waveforms }
  function  GetTimer0:word;
  { CCD operations performed by PCSG }
  procedure SDclearPCSG(Ntimes:word);
  procedure StoreS1PCSG(Ntimes:word);
  procedure StoreS2PCSG(Ntimes:word);
  procedure ShiftAlines1_3_PCSG(NL:word);
  procedure ShiftBlines1_3_PCSG(NL:word);
  procedure Transf1PCSG;
  procedure Transf2PCSG;
  procedure Transf3PCSG;
  procedure Shift1_3inAtr2_PCSG;
  procedure Shift1_3inBtr2_PCSG;
  procedure ABGClockPCSG(TimeMs:word);
  { $IfNDef REMTST}
  function  GetADCwait_PCSG: word;
  procedure GetPixADCwait_PCSG(var DataMin: TDataMinSt; WinW: word);
  { $EndIF}
  function  SDclearPCSGstop(Npix:word; StpAtN: word; var StpAtp:TstpAtpoint):word;
  procedure ClearIandS_PCSG(Ntimes:word);
  procedure ClearIandSandSR_PCSG(Ntimes:word);
  procedure CheckRAMn( WFnum: word );
  procedure PCSGTest1(Ntimes:word);
  procedure TestPCSGasm(NN:word);



implementation

 { $IfNDef REMTST}

uses
  StAqs01, StAQ01;

  { $ELSE}
  {
  procedure SetABGI_ON;
    begin
    end;

  procedure SetABGI_OFF;
    begin
    end;
```

```
  }
  { $EndIF}

const
  PCSG_Debug_ON     = false;

  { Timing and memory paratiters }

  LenghtWaveFormMAX =   63;
  WaveFormNumMax    =   64;
  PCSG_Sys_Clk_Dev  =    {2;} {3;}   {4;}  { 2;}  {6;} 10;

  { I/O addresses }

  { PC bus addresses      }
  { $IfNDef REMOTE}
  {
  PCSG_Timer0       = $380;
  PCSG_Timer1       = $382;
  PCSG_Timer2       = $384;
  PCSG_TimerContr   = $386;
  PCSG_StatusPort   = $3A0;
  PCSG_IACK         = $3A2;
  PCSG_RAM_RD       = $3A4;
  PCSG_RAM_WR       = $3A6;
  PCSG_WSR_Port     = $3A8;
  PCSG_ControlPort  = $3AA;
  }
  { $ELSE}
  PCSG_Timer0       = $1780;
  PCSG_Timer1       = $1782;
  PCSG_Timer2       = $1784;
  PCSG_TimerContr   = $1786;
  PCSG_StatusPort   = $17A0;
  PCSG_IACK         = $17A2;
  PCSG_RAM_RD       = $17A4;
  PCSG_RAM_WR       = $17A6;
  PCSG_WSR_Port     = $17A8;
  PCSG_ControlPort  = $17AA;
  { $EndIF}

  { PCSG_ControlPort Bits }

  PCSG_CR_RAMprogr    = $1;  { Program RAM }
  PCSG_CR_Latch_C     = $2;  { Enable V signal and control signal for PCSG }
  PCSG_CR_Spare       = $4;  { Spare }
  PCSG_CR_T01_EN      = $8;
  PCSG_CR_STOP_EN     = $10;
  PCSG_CR_H_EN        = $20; { 0 = high Z on H }
  PCSG_CR_T2_EN       = $40;
  PCSG_CR_WAIT_En     = $80;

  PCSG_ContrDisable=  $0;
  PCSG_RAM_Progr    =  PCSG_CR_RAMprogr;  { $1;}
  PCSG_RUN_V        =  PCSG_CR_Latch_C or PCSG_CR_T01_EN; { $2 or  $8; }
  PCSG_RUN_HV       =  PCSG_CR_Latch_C or PCSG_CR_T01_EN or PCSG_CR_H_EN; { $2 or  $8 or $20; }
  PCSG_ByPass       =  PCSG_ContrDisable;

  { PCSG_TimerContr bytes       }
  PCSG_T0_Control   = $30;      { bits 7,6=00   counter 0
```

```
                                  bits 5,4=11   16-bit mode
                                  bits 3-1=000  mode 0: interrupt on tc
                                  bit  0  =0    binary }
PCSG_T0_Status    = $C2;
PCSG_T1_Control   = $76;        { bits 7,6=01   counter 1
                                  bits 5,4=11   16-bit mode
                                  bits 3-1=011  mode 1: square wave
                                  bit  0  =0    binary }
PCSG_T1_Status    = $C4;

{ PCSG_StatusPort bits }

PCSG_SP_Stoped     = 1;
PCSG_SP_TimeOut_T0 = 2;
PCSG_SP_NoWAIT     = 4;
PCSG_SP_SecInc     = 8;

{ Signals for PCSG }

PCSG_IntPolPos    =    $1;
PCSG_IntSample    =    $2;
PCSG_IntReset     =    $4;
PCSG_AD_CS        =    $8;

PCSG_SRG1         =   $10;
PCSG_SRG2         =   $20;
PCSG_SRG3         =   $40;
PCSG_ABGC         =   $80;

PCSG_IAG          =  $100;
PCSG_TMG          =  $200;
PCSG_SAG1         =  $400;
PCSG_SAG2         =  $800;
PCSG_TRG          = $1000;
PCSG_SRGs         = $2000;

{ Sequencer instructions - loaded in RAM}

PCSG_Decr         = $4000;
PCSG_Cont         = $8000;

{ Signals for PCSG compound }

PCSG_Drn          = $0;
PCSG_DrnR         = PCSG_IntReset;

{ WaveForms numbers }

PCSG_ZeroWF           =  0;
PCSG_SerialShift      =  1;
PCSG_ClearSAG1        =  2;
PCSG_ClearSAG2        =  3;
PCSG_StoreSAG1        =  4;
PCSG_StoreSAG2        =  5;
PCSG_PshiftA          =  6;
PCSG_PshiftB          =  7;
PCSG_Transf1          =  8;
PCSG_Transf2          =  9;
PCSG_Transf3          = 10;
PCSG_Shift1_3inAtr2   = 11;
```

```
  PCSG_Shift1_3inBtr2   = 12;
  PCSG_GetADCwait       = 13;
  PCSG_GetPixADCwait    = 14;
  PCSG_ABGC_WF          = 15;

  PCSG_MaxWF            = 15;


type
  T_PCSG_WaveForm  = array[0..LenghtWaveFormMAX] of word;
  T_PCSG_Addrs     = array[1..2] of word;
  T_PCSG_WaveForms = array[0..WaveFormNumMax] of T_PCSG_WaveForm;
  T_PCSG_WaveAddrs = array[0..WaveFormNumMax] of T_PCSG_Addrs;

var
  PCSG_WaveForms: T_PCSG_WaveForms;
  PCSG_WAveAddrs: T_PCSG_WaveAddrs;

procedure PCSGloadSRAM(StartAddr, EndAddr:word; WaveForm:T_PCSG_WaveForm);
  var
    addr: word;
  begin
  { Required WaveForm Format }
  { WF[0] :=0;               }
  { WF[1] :=0;               }
  { WF[2] :=0; STOPS here    }
  { WF[2<n=EndAddr-StartAddr] = usefull data }
  {                                          }
  Port[PCSG_ControlPort]:=PCSG_RAM_Progr;  { SRAM for program }
  PortW[PCSG_WSR_Port]:=StartAddr;
  PortW[PCSG_RAM_WR]  :=WaveForm[StartAddr-StartAddr]
                        or PCSG_Cont or PCSG_Decr;
  for addr:=StartAddr+1 to EndAddr-1 do
    begin
    PortW[PCSG_WSR_Port]:=addr;
    PortW[PCSG_RAM_WR]  :=(word(WaveForm[addr-StartAddr]) or PCSG_Cont) and
                               not PCSG_Decr;
    end;
  PortW[PCSG_WSR_Port]:=EndAddr;
  PortW[PCSG_RAM_WR]  :=(WaveForm[EndAddr-StartAddr] or PCSG_Decr) and
                               not PCSG_Cont;
  PortW[PCSG_WSR_Port]:=StartAddr;
  Port[PCSG_ControlPort]:=PCSG_ContrDisable;
  end;

procedure CheckRAM(StartAddr, EndAddr:word; var WaveForm: T_PCSG_WaveForm);
  var
    addr: word;
  begin
  Port[PCSG_ControlPort]:=PCSG_RAM_Progr;  { SRAM for program }
  for addr:=StartAddr to EndAddr do
    begin
    PortW[PCSG_WSR_Port]:=addr;
    WaveForm[addr-StartAddr]:=PortW[PCSG_RAM_RD];
    end;
  Port[PCSG_ControlPort]:=PCSG_ContrDisable;
  end;

function ToBin(n:byte):string;
  var
```

```
    i:byte;
    st:string;
  begin
  st:='00000000';
  for i:=0 to 7 do
    begin
    if (n shr i) and 1 <> 0 then st[8-i]:= '1' ;
    end;
  ToBin:=st;
  end;

procedure CheckRAMwrite(SA, EA:word);
  var
    WF2: T_PCSG_WaveForm;
    i:word;
  begin
  CheckRAM(SA,EA,WF2);
  writeln('ADDR DEC         CD            ');
  {        I ii xxxxxx  00000000  00000000}
  for i:=0 to EA-SA do
    writeln('I ',i+SA:6,' ',WF2[i]:6, '  ',ToBin(Hi(WF2[i])),
                                  '  ',ToBin(Lo(WF2[i])) );
  end;

procedure CheckRAMn( WFnum: word );
  begin
  writeln(' Wave Form #: ', WFnum);
  CheckRAMwrite(PCSG_WaveAddrs[WFNum][1], PCSG_WaveAddrs[WFNum][2] );
  end;

procedure Set_Timer1(Frdev:word);
  begin
  {Port[PCSG_ControlPort]:=PCSG_ContrDisable;}
  Port[PCSG_TimerContr] :=PCSG_T1_Control;
  Port[PCSG_Timer1]     :=Lo(Frdev);
  Port[PCSG_Timer1]     :=hi(Frdev);
  end;

procedure CheckT1;
  var
    ch:char;
    Lt, Ht: word;
  begin
  Port[PCSG_ControlPort]:=PCSG_ContrDisable;
  Port[PCSG_TimerContr] :=PCSG_T1_Status;
  writeln(' T1 status ', Port[PCSG_Timer1]);
  Lt:= Port[PCSG_Timer1];
  Ht:= Port[PCSG_Timer1];
  writeln(' T1 counts ',Lt+256*Ht);
  end;

procedure Set_Timer0(Rep:word);
  begin
  {Port[PCSG_ControlPort]:=PCSG_ContrDisable;}
  Port[PCSG_TimerContr] :=PCSG_T0_Control;
  Port[PCSG_Timer0]     :=Lo(Rep);
  Port[PCSG_Timer0]     :=Hi(Rep);
  end;

procedure CheckT0;
```

```
  var
    ch:char;
    Lt, Ht: word;
  begin
  Port[PCSG_ControlPort]:=PCSG_ContrDisable;
  Port[PCSG_TimerContr] :=PCSG_T0_Status;
  writeln(' T0 status ', Port[PCSG_Timer0]);
  Lt:= Port[PCSG_Timer0];
  Ht:= Port[PCSG_Timer0];
  writeln(' T0 counts ',Lt+256*Ht );
  end;

function GetTimer0:word;
  var
    ch:char;
    Lt, Ht: word;
  begin
  {Port[PCSG_ControlPort]:=PCSG_ContrDisable;}
  Port[PCSG_TimerContr] :=PCSG_T0_Status;
  Lt:= Port[PCSG_Timer0];
  Ht:= Port[PCSG_Timer0];
  GetTimer0:=Lt+256*Ht ;
  end;

procedure SetWF_ZeroWF(StartAddr:word);
  const
    WFNum=PCSG_ZeroWF;
    WFlenght =8;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=$0;
  PCSG_WaveForms[WFNum][ 4]:=$0;
  PCSG_WaveForms[WFNum][ 5]:=$0;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_SerialShift(StartAddr:word);
  const
```

```
    WFNum=PCSG_SerialShift;
    WFlenght =6;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=PCSG_IntReset; {SDnR;}
  PCSG_WaveForms[WFNum][ 1]:=PCSG_IntReset; {SDnR;}
  PCSG_WaveForms[WFNum][ 2]:=PCSG_IntReset; {SDnR;}
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG3+PCSG_IntReset; {SDG3pR;}
  PCSG_WaveForms[WFNum][ 4]:=PCSG_SRG1+PCSG_IntReset; {SDG1pR;}
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG2+PCSG_IntReset; {SDG2pR;}
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_ClearSAG1(StartAddr:word);
  const
    WFNum=PCSG_ClearSAG1;
    WFlenght =5;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_TMG+PCSG_SAG1+
                             PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=$0;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
```

```
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_ClearSAG2(StartAddr:word);
  const
    WFNum=PCSG_ClearSAG2;
    WFlenght =5;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_TMG+PCSG_SAG2+
                             PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=$0;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_StoreSAG1(StartAddr:word);
  const
    WFNum=PCSG_StoreSAG1;
    WFlenght =10;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_TMG+PCSG_SAG1+
                             PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_TMG+PCSG_SAG1+
                             PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_TMG+PCSG_SAG1+
```

```
                                PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 9]:=PCSG_IAG;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_StoreSAG2(StartAddr:word);
  const
    WFNum=PCSG_StoreSAG2;
    WFlenght =10;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_TMG+PCSG_SAG2+
                                PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_TMG+PCSG_SAG2+
                                PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_TMG+PCSG_SAG2+
                                PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 9]:=PCSG_IAG;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_PshiftA(StartAddr:word);
  const
    WFNum=PCSG_PshiftA;
    WFlenght =5;
```

```
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SAG1+
                              PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=$0;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_PshiftB(StartAddr:word);
  const
    WFNum=PCSG_PshiftB;
    WFlenght =5;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SAG2+
                              PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=$0;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
```

```
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_Transf1(StartAddr:word);
  const
    WFNum=PCSG_Transf1;
    WFlenght =6;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 6]:=$0;
  PCSG_WaveForms[WFNum][ 7]:=$0;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_Transf2(StartAddr:word);
  const
    WFNum=PCSG_Transf2;
    WFlenght =8;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 8]:=$0;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
```

```
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_Transf3(StartAddr:word);
  const
    WFNum=PCSG_Transf3;
    WFlenght =9;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_TRG;
  PCSG_WaveForms[WFNum][ 9]:=$0;
  PCSG_WaveForms[WFNum][10]:=$0;
  PCSG_WaveForms[WFNum][11]:=$0;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_Shift1_3inAtr2(StartAddr:word);
  const
    WFNum=PCSG_Shift1_3inAtr2;
    WFlenght =12;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 1]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 2]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
```

```
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_SAG1+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 9]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][10]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][11]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;


procedure SetWF_Shift1_3inBtr2(StartAddr:word);
  const
    WFNum=PCSG_Shift1_3inBtr2;
    WFlenght =12;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 1]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 2]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_SAG2+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 9]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][10]:=PCSG_TRG+PCSG_IntReset;
  PCSG_WaveForms[WFNum][11]:=PCSG_SRG1+PCSG_SRG2+PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][12]:=$0;
  PCSG_WaveForms[WFNum][13]:=$0;
  PCSG_WaveForms[WFNum][14]:=$0;
  PCSG_WaveForms[WFNum][15]:=$0;
  PCSG_WaveForms[WFNum][16]:=$0;
  PCSG_WaveForms[WFNum][17]:=$0;
  PCSG_WaveForms[WFNum][18]:=$0;
  PCSG_WaveForms[WFNum][19]:=$0;
  PCSG_WaveForms[WFNum][20]:=$0;
  PCSG_WaveForms[WFNum][21]:=$0;
  PCSG_WaveForms[WFNum][22]:=$0;
  PCSG_WaveForms[WFNum][23]:=$0;
  end;

procedure SetWF_GetADCwait(StartAddr:word);
  const
```

```
    WFNum=PCSG_GetADCwait;
    WFlenght =9;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=$0;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_AD_CS;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_AD_CS;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_AD_CS;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_AD_CS;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_AD_CS;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_AD_CS;
  end;

procedure SetWF_GetPixADCwait(StartAddr:word);
  const
    WFNum=PCSG_GetPixADCwait;
    WFlenght =26;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;

  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 3]:=PCSG_SRG3+PCSG_IntReset;
  PCSG_WaveForms[WFNum][ 4]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][ 5]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][ 6]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][ 7]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][ 8]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][ 9]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][10]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][11]:=PCSG_SRG1+PCSG_IntSample;
  PCSG_WaveForms[WFNum][12]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][13]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][14]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][15]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][16]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][17]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][18]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][19]:=PCSG_SRG2+PCSG_IntSample+PCSG_IntPolPos;
  PCSG_WaveForms[WFNum][20]:=PCSG_SRG2+PCSG_AD_CS;
  PCSG_WaveForms[WFNum][21]:=PCSG_SRG2+PCSG_AD_CS;
  PCSG_WaveForms[WFNum][22]:=PCSG_SRG2+PCSG_AD_CS;
  PCSG_WaveForms[WFNum][23]:=PCSG_SRG2+PCSG_AD_CS;
  PCSG_WaveForms[WFNum][24]:=PCSG_SRG2+PCSG_AD_CS;
  PCSG_WaveForms[WFNum][25]:=PCSG_SRG2+PCSG_AD_CS;
  end;

procedure SetWF_ABGC(StartAddr:word);
  const
    WFNum=PCSG_ABGC_WF;
    WFlenght =4;
  begin
  PCSG_WaveAddrs[WFNum][1]:=StartAddr;
  PCSG_WaveAddrs[WFNum][2]:=StartAddr+WFlenght-1;
```

```
  PCSG_WaveForms[WFNum][ 0]:=$0;
  PCSG_WaveForms[WFNum][ 1]:=$0;
  PCSG_WaveForms[WFNum][ 2]:=PCSG_ABGC;
  PCSG_WaveForms[WFNum][ 3]:=$0;
  end;

procedure LoadWF( WFNum: word );
  begin
  PCSGloadSRAM(PCSG_WaveAddrs[WFNum][1], PCSG_WaveAddrs[WFNum][2],
               PCSG_WaveForms[WFNum]);
  end;

procedure SetAndLoadWFs;
  var
    i: word;
  begin
  PCSG_ToByPass;
  Set_Timer1(PCSG_Sys_Clk_Dev);
  SetWF_ZeroWF(0);
  SetWF_SerialShift(PCSG_WaveAddrs[1-1][2]+1);
  SetWF_ClearSAG1(PCSG_WaveAddrs[2-1][2]+1);
  SetWF_ClearSAG2(PCSG_WaveAddrs[3-1][2]+1);
  SetWF_StoreSAG1(PCSG_WaveAddrs[4-1][2]+1);
  SetWF_StoreSAG2(PCSG_WaveAddrs[5-1][2]+1);
  SetWF_PshiftA(PCSG_WaveAddrs[6-1][2]+1);
  SetWF_PshiftB(PCSG_WaveAddrs[7-1][2]+1);
  SetWF_Transf1(PCSG_WaveAddrs[8-1][2]+1);
  SetWF_Transf2(PCSG_WaveAddrs[9-1][2]+1);
  SetWF_Transf3(PCSG_WaveAddrs[10-1][2]+1);
  SetWF_Shift1_3inAtr2(PCSG_WaveAddrs[11-1][2]+1);
  SetWF_Shift1_3inBtr2(PCSG_WaveAddrs[12-1][2]+1);
  SetWF_GetADCwait(PCSG_WaveAddrs[13-1][2]+1);
  SetWF_GetPixADCwait(PCSG_WaveAddrs[14-1][2]+1);
  SetWF_ABGC(PCSG_WaveAddrs[15-1][2]+1);
  for i:=0 to PCSG_MaxWF do LoadWF(i);
  PCSG_ToByPass;
  end;

  {
procedure TestToBin;
  var
    i,j,k:byte;
    ch:char;
  begin
  k:=0;
  for i:=0 to 51 do
    begin
    for j:=0 to 4 do
      begin
      write('N ',k:3,' ',ToBin(k),' ');
      if k<255 then inc(k);
      end;
    writeln;
    end;
  ch:=ReadKey;
  end;
  }

procedure PCSG_ToByPass;
```

```
  begin
  PortW[PCSG_ControlPort]:=PCSG_ByPass;
  end;

procedure ExecWFV( WFnum, Times: word );
  begin
  {ByPassOFF;}
  Set_Timer0(Times);
  PortW[PCSG_WSR_Port]    := PCSG_WaveAddrs[WFNum][1];
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN; { $8;}
  Port[PCSG_ControlPort] := PCSG_RUN_V;
  Port[PCSG_IACK]:=0;
  end;

procedure ExecWF( WFnum, Times: word );
  begin
  {ByPassOFF;}
  Set_Timer0(Times);
  PortW[PCSG_WSR_Port]    := PCSG_WaveAddrs[WFNum][1];
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN or PCSG_CR_RAMprogr;
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN; { $8;}
  Port[PCSG_ControlPort] := PCSG_RUN_HV;
  Port[PCSG_IACK]:=0;
  end;

procedure ExecWFwait( WFnum, Times: word );
  begin
  {ByPassOFF;}
  Set_Timer0(Times);
  PortW[PCSG_WSR_Port]    := PCSG_WaveAddrs[WFNum][1];
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN or PCSG_CR_RAMprogr;
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN; { $8;}
  Port[PCSG_ControlPort] := PCSG_RUN_HV or PCSG_CR_WAIT_En;
  Port[PCSG_IACK]:=0;
  end;

procedure ExecWFstop( WFnum, Times: word );
  begin
  {ByPassOFF;}
  Set_Timer0(Times);
  PortW[PCSG_WSR_Port]    := PCSG_WaveAddrs[WFNum][1];
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN or PCSG_CR_RAMprogr;
  PortW[PCSG_ControlPort]:= PCSG_CR_T01_EN; { $8;}
  Port[PCSG_ControlPort] := PCSG_RUN_HV or PCSG_CR_STOP_En;
  Port[PCSG_IACK]:=0;
  end;

procedure Wait_PCSG_Finish;
  begin
  repeat
    until Port[PCSG_StatusPort] and (PCSG_SP_TimeOut_T0 or PCSG_SP_SecInc) =
        PCSG_SP_TimeOut_T0;
  end;

function  PCSG_Finish:boolean;
  begin
  PCSG_Finish := Port[PCSG_StatusPort] and (PCSG_SP_TimeOut_T0 or PCSG_SP_SecInc) =
        PCSG_SP_TimeOut_T0;
  end;
```

```
function  SDclearPCSGstop(Npix:word; StpAtN: word; var StpAtp:TstpAtpoint):word;
  var
    Nst, T0, StsP: word;
    Lt, Ht: byte;
  begin
  Nst:=0;
  SDclearPCSGstop:=Nst;
  if Npix=0 then EXIT;
  ExecWFstop(PCSG_SerialShift, Npix);
  repeat
    begin
    StsP:=Port[PCSG_StatusPort];
    if StsP and PCSG_SP_Stoped = PCSG_SP_Stoped then
      begin
      {GenSynch;}
      {T0:=GetTimer0;}
      Lt:=Port[PCSG_Timer0];
      Ht:=Port[PCSG_Timer0];
      T0:=Lt+256*Ht;
      if Nst<=StpAtN then
        begin
        inc(Nst);
        StpAtp[Nst]:=Npix-T0+PixNotLook;
        end
      else
        begin
        Port[PCSG_ControlPort]:=PCSG_Run_HV;
        end;
      if StsP and PCSG_SP_TimeOut_T0 <> PCSG_SP_TimeOut_T0 then
        begin
        Port[PCSG_IACK]:=0;
        end;
      end;
    end;
    until StsP and (PCSG_SP_TimeOut_T0 or PCSG_SP_SecInc) = PCSG_SP_TimeOut_T0;
  StpAtp[0]:=Nst;
  SDclearPCSGstop:=Nst;
  end;

procedure SDclearPCSG(Ntimes:word);
  begin
  if Ntimes=0 then EXIT;
  ExecWF(PCSG_SerialShift, Ntimes);
  Wait_PCSG_Finish;
  end;

procedure StoreS1PCSG(Ntimes:word);
  begin
  if Ntimes=0 then EXIT;
  SetABGI_ON;
  ExecWF(PCSG_ClearSAG1, 1);
  Wait_PCSG_Finish;
  ExecWF(PCSG_StoreSAG1, Ntimes);
  Wait_PCSG_Finish;
  ExecWF(PCSG_ClearSAG1, 5);
  Wait_PCSG_Finish;
  SetABGI_OFF;
  end;

procedure StoreS2PCSG(Ntimes:word);
```

```
  begin
  if Ntimes=0 then EXIT;
  SetABGI_ON;
  ExecWF(PCSG_ClearSAG2, 1);
  Wait_PCSG_Finish;
  ExecWF(PCSG_StoreSAG2, Ntimes);
  Wait_PCSG_Finish;
  ExecWF(PCSG_ClearSAG2, 5);
  Wait_PCSG_Finish;
  SetABGI_OFF;
  end;

procedure ShiftAlines1_3_PCSG(NL:word);
  begin
  if NL=0 then EXIT;
  ExecWF(PCSG_PshiftA, NL);
  Wait_PCSG_Finish;
  end;

procedure ShiftBlines1_3_PCSG(NL:word);
  begin
  if NL=0 then EXIT;
  ExecWF(PCSG_PshiftB, NL);
  Wait_PCSG_Finish;
  end;

procedure Transf1PCSG;
  begin
  ExecWF(PCSG_Transf1, 1);
  Wait_PCSG_Finish;
  end;

procedure Transf2PCSG;
  begin
  ExecWF(PCSG_Transf1, 1);
  Wait_PCSG_Finish;
  end;

procedure Transf3PCSG;
  begin
  ExecWF(PCSG_Transf1, 1);
  Wait_PCSG_Finish;
  end;

procedure Shift1_3inAtr2_PCSG;
  begin
  ExecWF(PCSG_Shift1_3inAtr2, 1);
  Wait_PCSG_Finish;
  end;

procedure Shift1_3inBtr2_PCSG;
  begin
  ExecWF(PCSG_Shift1_3inBtr2, 1);
  Wait_PCSG_Finish;
  end;

procedure ABGClockPCSG(TimeMs:word);
  const
    ABGCmul=500;
  var
```

```
    rep, Tmod, i: word;
  begin
  if TimeMs<64 then
    begin
    Tmod:=TimeMs*ABGCmul;
    if TimeMs=0 then Tmod:=10;
    ExecWF(PCSG_ABGC_WF, Tmod);
    Wait_PCSG_Finish;
    end
  else
    begin
    rep := TimeMs div 50;
    if rep<1 then rep:=1;
    if TimeMs-rep*50<0 then Tmod:=0 else Tmod:=TimeMs-rep*50;
    for i:=1 to rep do
      begin
      ExecWF(PCSG_ABGC_WF, 50*ABGCmul);
      Wait_PCSG_Finish;
      end;
    if (Tmod>0) and (Tmod<50) then
      begin
      ExecWF(PCSG_ABGC_WF, Tmod*ABGCmul);
      Wait_PCSG_Finish;
      end;
    end;

  end;

{ $IfNDef REMTST}

function  GetADCwait_PCSG: word;
  var
    DDD: word;
  begin
  {GenSynch;}
  {$IfNDef REMOTE }

  ASM
    mov  dx, ADCstr
    mov  ax, AD_CSr
    out  dx, ax
    mov  ax, ADCstrInit
    out  dx, ax
    end;   {ASM}
  {$EndIf }
  ExecWFwait(PCSG_GetADCwait, 1);
  {$IfNDef REMOTE }
  ASM
    mov  cx, 10
    mov  si, ADC_BUSY_0 or ADC_M_WAIT_0
    mov  dx, ADCouts
    @1:
    in   ax, dx
    mov  bx, ax
    and  bx, si
    cmp  bx, ADC_BUSY_0
    loopne @1
    mov  DDD, ax
    mov  dx, ADCstr
    mov  ax, AD_CSr
```

```
    out  dx, ax
    mov  ax, ADCstrInit
    out  dx, ax
    end;   {ASM}
  {Wait_PCSG_Finish;}
  {$Else }
  ASM
    mov  cx, 10
    mov  si, ADC_BUSY_O or ADC_M_WAIT_O
    mov  dx, ADCouts
    @1:
    in   ax, dx
    mov  bx, ax
    and  bx, si
    cmp  bx, ADC_BUSY_O
    loopne @1
    mov  DDD, ax
    {******************}
    {To be removed }
    mov  dx, ADCstr
    mov  ax, AD_CSr
    out  dx, ax
    mov  ax, ADCstrInit
    out  dx, ax
    {*******************}
    end;   {ASM}
  {$EndIf }
  GetADCwait_PCSG:=DDD;
  end;

procedure GetPixADCwait_PCSG(var DataMin: TDataMinSt; WinW: word);
  begin
  {$IfNDef REMOTE }

  Port[ADCstr]:=AD_CSr;
  Port[ADCstr]:=ADCstrInit;

  {$EndIf }
  ExecWFwait(PCSG_GetPixADCwait, WinW);
  {GenSynch;}
  {
  ASM
    lds  di, DataMin
    mov  cx, WinW
    mov  si, ADC_M_WAIT_O
    mov  bx, ADC_BUSY_O
    @2:
    mov  dx, ADCouts
    @1:
    @3:
    in   ax, dx
    xor  ax, si
    test ax, si
    jz   @1
    test ax, bx
    jz   @3
    mov  [ds:di], ax
    inc  di
    inc  di
    mov  dx, ADCstr
```

```
  mov  ax, AD_CSr
  out  dx, ax
  mov  ax, ADCstrInit
  out  dx, ax
  loop @2
  end; }  {ASM}

{$IfNDef REMOTE }
ASM
  push bp
  lds  di, DataMin
  mov  cx, WinW
  mov  si, ADC_BUSY_0 or ADC_M_WAIT_0
  @2:
  mov  bp, 10
  mov  dx, ADCouts
  xchg bp, cx
  @1:
  in   ax, dx
  mov  bx, ax
  and  bx, si
  cmp  bx, ADC_BUSY_0
  loopne @1
  xchg bp, cx
  mov  [ds:di], ax
  inc  di
  inc  di
  mov  dx, ADCstr
  mov  ax, AD_CSr
  out  dx, ax
  mov  ax, ADCstrInit
  out  dx, ax
  loop @2
  pop  bp
  end;   {ASM}
{$Else }
ASM
  push bp
  lds  di, DataMin
  mov  cx, WinW
  mov  si, ADC_BUSY_0 or ADC_M_WAIT_0
  @2:
  mov  bp, 10
  mov  dx, ADCouts
  xchg bp, cx
  @1:
  in   ax, dx
  mov  bx, ax
  and  bx, si
  cmp  bx, ADC_BUSY_0
  loopne @1
  xchg bp, cx
  mov  [ds:di], ax
  inc  di
  inc  di
  {****************}
  {To be removed }
  mov  dx, ADCstr
  mov  ax, AD_CSr
  out  dx, ax
```

```
    mov  ax, ADCstrInit
    out  dx, ax
    {*******************}
    loop @2
    pop  bp
    end;   {ASM}
  {$EndIf }
  end;

{ $EndIF}

procedure ClearIandS_PCSG(Ntimes:word);
  begin
  if Ntimes>0 then
    begin
    SetABGI_ON;
    ExecWF(PCSG_StoreSAG1, Ntimes*(NofA2lines+10));
    Wait_PCSG_Finish;
    ExecWF(PCSG_StoreSAG2, Ntimes*(NofA2lines+10));
    Wait_PCSG_Finish;
    SetABGI_OFF;
    end;
  {ByPassON;}
  {Transf3;}
  end;

procedure ClearIandSandSR_PCSG(Ntimes:word);
  begin
  if Ntimes>0 then
    begin
    SetABGI_ON;
    ExecWF(PCSG_StoreSAG1, Ntimes*(NofA2lines+10));
    Wait_PCSG_Finish;
    ExecWF(PCSG_StoreSAG2, Ntimes*(NofA2lines+10));
    Wait_PCSG_Finish;
    SetABGI_OFF;
    end;
  {ByPassON;}
  {Transf3;}
  SDclearPCSG(NofSelements+NofDLpixels);
  end;

procedure PCSG_Initialize;
  var
    ch:char;
    i: word;
  begin
  SetAndLoadWFs;
  {
  if PCSG_Debug_On then
    begin
    ClrScr;
    writeln('PCSG memory TEST');
    writeln;
    for i:=0 to PCSG_MaxWF do
      begin
      CheckRAMn(i);
      if i<>PCSG_MaxWF then writeln('Press any Key to see next WaveForm')
                       else writeln('Press any Key to START PCSG');
      ch:=Readkey;
```

```
    end;
  writeln('StoreS1PCSG - TEST');
  writeln('Press any Key to Continue');
  repeat
    GenSynch;
    StoreS1PCSG(10);
    until KeyPressed;
  ch:=ReadKey;

  writeln('StoreS2PCSG - TEST');
  writeln('Press any Key to Continue');
  repeat
    GenSynch;
    StoreS2PCSG(10);
    until KeyPressed;
  ch:=ReadKey;

  PCSG_TOByPass;

  writeln;
  writeln('Press any key to EXIT');
  ch:=ReadKey;

  end;
}

end;

procedure PCSGTest1(Ntimes:word);
  begin
  ExecWF(PCSG_Transf1, Ntimes);
  end;

procedure TestPCSGasm(NN:word);
  var
    j: word;
  begin
  for j:=0 to NN do
    begin
    Port[PCSG_TimerContr] :=PCSG_TO_Status;
    end;
  end;

Begin
  PCSG_Initialize;
End.
```

```
unit StTest;

{ Housekiping and display Search result - in TEST mode }

interface

procedure TestV(Times: longInt);
{$IfNDef REMOTE }
procedure ShowFastSearch(Np:integer);
{$EndIf }


implementation

uses
  {$IfNDef REMOTE }
  CRT,
  {$EndIf }
  Stdef01, StAqs01, StAQ01, STMain01, StPpic01, StCpic04,
  StAux01, StPCSG01, StSrch01, StRem1;


{$IfNDef REMOTE }
procedure ShowFastSearch(Np:integer);
  var
    j, i, Jm: word;
    ch: char;
    Lfr: word;
  begin
  ClrScr;
  writeln(' Search for line  ', LineNotLook+1:3, ' to 976 only odd lines');
  writeln(' Search for pixel ', PixNotLook:3,    ' to 1194; 1170 last light');
  writeln(' DAC treshold ', DAC3V:4, ' Exposure ',ExposureTime,
          ' Total pix above treshold ',Np);
  writeln;
  writeln('Line # pix                            X coordinate');
  writeln;
  if true {StpAtpoints[0,0]>0} then
    begin
    if StpAtlinesMax<=40 then Lfr:=StpAtlinesMax else Lfr:=40;
    for j:= {StpAtlinesMax} Lfr downto 1 do
      begin
      write(StpAtpoints[j,0]:3,' ');
      write(StpAtpoints[j,StpAtpointsMax+1]:4,' ');
      for i:=1 to {3*stpAtN} 14 do
        begin
        write(StpAtpoints[j,i]:4,' ');
        end; {for}
        writeln;
      end; {for}
    end; {if}
  if StarSearch.StNfound>0 then
    begin
    writeln('Star Found ',StarSearch.StNfound);
    for j:=1 to StarSearch.StNfound do
      begin
      write('Star ',j:2,' LineB ',StarSearch.StSearchDA[j].LineB,
                      ' LineM ',StarSearch.StSearchDA[j].LineM,
                      ' LineT ',StarSearch.StSearchDA[j].LineT);
      write(          ' PixL ',StarSearch.StSearchDA[j].PixL,
```

```
                        ' PixM ',StarSearch.StSearchDA[j].PixM,
                        ' PixR ',StarSearch.StSearchDA[j].PixR);
      writeln;
      end;
    end;
  {
  write('To Exit search press ESC; < > change treshold; space to cycle treshold');

  writeln('Press any Key to Continue');
  repeat
    ClearIandSandSR_PCSG(2);
    until Keypressed;
  ch:=ReadKey;
  }
  end;
{$EndIf }

{$IfNDef REMOTE }
function DisplTestV: char;
  var
    i: integer;
  const
    Factor: array[0..VoltMax] of float =
      (1.0, 0.32, 0.32, 0.5, 0.5, 0.5, 0.5, 1.0,
       0.5, 0.5,  0.5,  0.5, 0.5, 0.32,0.32,0.32,
       1.0, 1.0,  1.0,  1.0, 1.0, 1.0, 1.0, 1.0 ,
       1.0);
  begin
  ClrScr;
  GoToXY(1,1);
  writeln('Internal Voltage Test');
  for i:=0 to 23 do
        begin
        case i of
           0: write('Ch ',i:2,' TempOut           ');
           1: write('Ch ',i:2,' +12V c            ');
           2: write('Ch ',i:2,' +12V a            ');
           3: write('Ch ',i:2,' +6VDr (+5)        ');
           4: write('Ch ',i:2,' +5V c             ');
           5: write('Ch ',i:2,' +5V a             ');
           6: write('Ch ',i:2,' +2V               ');
           7: write('Ch ',i:2,' -Bias (-4.6V)     ');
           8: write('Ch ',i:2,' -ABG (-7V)        ');
           9: write('Ch ',i:2,' mIAG (-5.5V)   D2 ');
          10: write('Ch ',i:2,' mABG (-1.5V)      ');
          11: write('Ch ',i:2,' IntBofs(-1.5V) D1 ');
          12: write('Ch ',i:2,' +Bias (OV)        ');
          13: write('Ch ',i:2,' -11V              ');
          14: write('Ch ',i:2,' -12V a            ');
          15: write('Ch ',i:2,' -12V c            ');
          16: write('Ch ',i:2,' Spare             ');
          17: write('Ch ',i:2,' VoltMonit         ');
          18: write('Ch ',i:2,' Amp1Out           ');
          19: write('Ch ',i:2,' CCD direct        ');
          20: write('Ch ',i:2,' Temp PC           ');
          21: write('Ch ',i:2,' IntBout           ');
          22: write('Ch ',i:2,' Temp Lens         ');
          23: write('Ch ',i:2,' Temp CCD          ');
          end; {case}
        writeln(' ADC ',($FFF and Voltages[i]):4,' = ',
```

```
              ( ($FFF and Voltages[i])-2048.0)/2048.0 * 5.0 / factor[i] :6:2,' V ' );
        end;
  writeln('Press SPACE to start acquisition, ESC to EXIT');
  delay(1000);
  if keyPressed then DisplTestV:=ReadKey else displTestV:='x';
  end;
{$EndIf }

{$IfDef REMOTE }
function DisplTestV: char;
  var
    i: integer;
    St, St0, st1, st2, st3: string;
  const
    Factor: array[0..VoltMax] of float =
      (1.0, 0.32, 0.32, 0.5, 0.5, 0.5, 0.5, 1.0,
       0.5, 0.5,  0.5,  0.5, 0.5, 0.32,0.32,0.32,
       1.0, 1.0,  1.0,  1.0, 1.0, 1.0, 1.0, 1.0 ,
       1.0);
  begin
  PutStr('Internal Voltage Test'+#13#10);
  for i:=0 to 23 do
        begin
        Str(i:2,st1);
        case i of
           0: st2:=' TempOut          ';
           1: st2:=' +12V c           ';
           2: st2:=' +12V a           ';
           3: st2:=' +6VDr (+5)       ';
           4: st2:=' +5V c            ';
           5: st2:=' +5V a            ';
           6: st2:=' +2V              ';
           7: st2:=' -Bias (-4.6V)    ';
           8: st2:=' -ABG (-7V)       ';
           9: st2:=' mIAG (-5.5V)   D2 ';
          10: st2:=' mABG (-1.5V)     ';
          11: st2:=' IntBofs(-1.5V) D1 ';
          12: st2:=' +Bias (0V)       ';
          13: st2:=' -11V             ';
          14: st2:=' -12V a           ';
          15: st2:=' -12V c           ';
          16: st2:=' Spare            ';
          17: st2:=' VoltMonit        ';
          18: st2:=' Amp1Out          ';
          19: st2:=' CCD direct       ';
          20: st2:=' Temp PC          ';
          21: st2:=' IntBout          ';
          22: st2:=' Temp Lens        ';
          23: st2:=' Temp CCD         ';
          end; {case}
        St:='Ch '+st1+st2;
        str(($FFF and Voltages[i]):4,st1);
        str(( ($FFF and Voltages[i])-2048.0)/2048.0 * 5.0 / factor[i] :6:2, st2);
        St0:=' ADC '+st1+' = '+st2;
        St:=St+St0;
        PutStr(St+#13#10);
        Delay(500);
        {
        writeln(' ADC ',($FFF and Voltages[i]):4,' = ',
              ( ($FFF and Voltages[i])-2048.0)/2048.0 * 5.0 / factor[i] :6:2,' V ' );
```

```
      }
      end;
  PutStr('Press SPACE to start acquisition, ESC to EXIT'+#13#10);
  delay(1000);
  DisplTestV:=GetChar;
  end;
{$EndIf }

procedure TestV(Times: longInt);
  var
    i: byte;
    j: longInt;
    ch: char;
  const
    dirDAC0: boolean = true;
    dirDAC1: boolean = false;
    dirDAC2: boolean = true;
    dirDAC3: boolean = false;
    dirDAC4: boolean = true;
    dirDAC5: boolean = false;
    dirDAC6: boolean = true;
    dirDAC7: boolean = false;
  begin
  j:=0;
  ClearIandSandSR_PCSG(2);
  repeat
    begin
    inc(j);
    ClearIandSandSR_PCSG(1);
    if J>20 then PCooler_ON;
    {
    if dirDAC0 then inc(DAC0V,100) else dec(DAC0V,100);
    if DAC0V>=3990 then dirDAC0:=false;
    if DAC0V<=120  then dirDAC0:=true;
    if dirDAC1 then inc(DAC1V,100) else dec(DAC1V,100);
    if DAC1V>=3990 then dirDAC1:=false;
    if DAC1V<=120  then dirDAC1:=true;
    if dirDAC2 then inc(DAC2V,100) else dec(DAC2V,100);
    if DAC2V>=3990 then dirDAC2:=false;
    if DAC2V<=120  then dirDAC2:=true;
    if dirDAC3 then inc(DAC3V,100) else dec(DAC3V,100);
    if DAC3V>=3990 then dirDAC3:=false;
    if DAC3V<=120  then dirDAC3:=true;
    if dirDAC4 then inc(DAC4V,100) else dec(DAC4V,100);
    if DAC4V>=3990 then dirDAC4:=false;
    if DAC4V<=120  then dirDAC4:=true;
    if dirDAC5 then inc(DAC5V,100) else dec(DAC5V,100);
    if DAC5V>=3990 then dirDAC5:=false;
    if DAC5V<=120  then dirDAC5:=true;
    if dirDAC6 then inc(DAC6V,100) else dec(DAC6V,100);
    if DAC6V>=3990 then dirDAC6:=false;
    if DAC6V<=120  then dirDAC6:=true;
    if dirDAC7 then inc(DAC7V,100) else dec(DAC7V,100);
    if DAC7V>=3990 then dirDAC7:=false;
    if DAC7V<=120  then dirDAC7:=true;
    }
    Anal_ON;
    LoadDAC(DAC0V, 0);
    LoadDAC(DAC1V, 1);
    LoadDAC(DAC2V, 2);
```

```
    LoadDAC(DAC3V, 3);
    LoadDAC(DAC4V, 4);
    LoadDAC(DAC5V, 5);
    LoadDAC(DAC6V, 6);
    LoadDAC(DAC7V, 7);
    ClearIandSandSR_PCSG(1);
    VoltMonit(Voltages);
    ClearIandSandSR_PCSG(1);
    {$IfNDef REMOTE }
    ch:=DisplTestV;
    {$Else }
      {$IfNDef REMS}
      CH:=' ';
      {$ELSE}
      ch:=DisplTestV;
      {$EndIf}
    {$eNDiF }
    ClearIandSandSR_PCSG(1);
    end;
    until (ch=#27) or (ch=' ') or (ch=#0) or (j>=Times);
  if ch=#27 then HALT;
  end;

Begin
End.
```

```
unit StSrch01;

{ Search control }

interface

uses
  StDef01;

function  StSearch(Nstar:word): word;

implementation

uses
  StAq01, StPCSG01, StPpic01, Sttest;

function  StLookPix(yi: word): boolean;
  var
    i, Xmin, Xmax: word;
  begin
  StLookPix:=false;
  if  (StpAtPoints[yi,StpAtPointsMax+1]<StpAtpointsMax) and
      (StpAtPoints[yi,StpAtPointsMax+1]>=1) then
    begin
    Xmax:=0;
    Xmin:=NofImageMaxX;
    for i:=1 to StpAtPoints[yi,StpAtPointsMax+1] do
      begin
      if StpAtPoints[yi,i]<Xmin then Xmin:=StpAtPoints[yi,i];
      if StpAtPoints[yi,i]>Xmax then Xmax:=StpAtPoints[yi,i];
      end;
    if (Xmax-Xmin>=PixWreq) and (Xmax-Xmin<PixWreqMax) then
      begin
      StLookPix:=true;
      end;
    end;
  end;

function  Xmean(yi: word; var Xmin,Xmax, XW: word): word;
  var
    i: word;
  begin
  Xmax:=0;
  Xmin:=NofImageMaxX;
  for i:=1 to StpAtPoints[yi,StpAtPointsMax+1] do
    begin
    if StpAtPoints[yi,i]<Xmin then Xmin:=StpAtPoints[yi,i];
    if StpAtPoints[yi,i]>Xmax then Xmax:=StpAtPoints[yi,i];
    end;
  XW:=Xmax-Xmin;
  Xmean:= round((Xmax+Xmin)/2);
  end;

function  StLook: word;
  var
    xi, yi: word;
    YgapFrom:word;
    LineFs, LineLs, Yf, Yl: word;
  begin
  StLook:=0;
```

```
StarSearch.StNsearch:=1;
StarSearch.StNfound:=0;
if StpAtpoints[0,1]>=9999 then EXIT;
StarSearch.BadLines:=0;
StarSearch.DAC3VTh:=DAC3V;
LineFs:=0;
LineLs:=0;
Yf:=0;
Yl:=0;
YgapFrom:=LineNotLook;
yi:=0;
for yi:=1 to StpAtPoints[0,0] do
  begin
  if stLookPix(yi) and
     ((StpAtPoints[yi,0]-YgapFrom)>=LineGapRequired) and
     ((NofPlines-StpAtPoints[yi,0])>=2*LineGapRequired) and
     ((StpAtPoints[yi,0]=LineLs+2) or (LineFs=0 ) ) and
     (yi<StpAtPoints[0,0]) then
    begin
    if LineFs=0 then
      begin
      LineFs:=StpAtPoints[yi,0];
      LineLs:=LineFs;
      Yf:=yi;
      Yl:=Yf;
      end
    else
      begin
      LineLs:=StpAtPoints[yi,0];
      Yl:=yi;
      end;
    end
  else
    begin
    if (yi=StpAtPoints[0,0]) and
       stLookPix(yi) and
       ((StpAtPoints[yi,0]-YgapFrom)>=LineGapRequired) and
       ((NofPlines-StpAtPoints[yi,0])>=2*LineGapRequired) and
       (StpAtPoints[yi,0]=LineLs+2) then
         if LineFs>0 then
           begin
           LineLs:=StpAtPoints[yi,0];
           Yl:=yi;
           end;

    if (LineFs>0) and
       ( (LineLs-LineFs)>=LineWreq ) and ((LineLs-LineFs)<=LineWreqMax) and
         ( (StpAtPoints[yi,StpAtPointsMax+1]<StpAtpointsMax) or
         (StpAtPoints[yi,0]>LineLs+LineGapRequired) or
          (yi=StpAtPoints[0,0])  )  then
        begin
        inc(StarSearch.StNfound);
        StarSearch.StSearchDA[StarSearch.StNsearch].LineB:=LineFs;
        StarSearch.StSearchDA[StarSearch.StNsearch].LineT:=LineLs;
        StarSearch.StSearchDA[StarSearch.StNsearch].LineM:=(LineLs+LineFs) div 2;
        StarSearch.StSearchDA[StarSearch.StNsearch].LineW:=(LineLs-LineFs);
        StarSearch.StSearchDA[StarSearch.StNsearch].LineGapB:=Linefs-YgapFrom;
        StarSearch.StSearchDA[StarSearch.StNsearch].LineGapT:=0;

        StarSearch.StSearchDA[StarSearch.StNsearch].PixW:=
```

```
            Xmean( (Yf+Yl) div 2,
            StarSearch.StSearchDA[StarSearch.StNsearch].PixL,
            StarSearch.StSearchDA[StarSearch.StNsearch].PixM,
            StarSearch.StSearchDA[StarSearch.StNsearch].PixR);

          if (StarSearch.StNsearch>1) and
             (StarSearch.StSearchDA[StarSearch.StNsearch-1].LineGapT=0) then
            begin
            StarSearch.StSearchDA[StarSearch.StNsearch-1].LineGapT:=LineFs-YgapFrom;
            end;
          YgapFrom:=LineLs;
          if stLookPix(yi)  then
            begin
            Linefs:=StpAtPoints[yi,0];
            LineLs:=LineFs;
            Yf:=yi;
            Yl:=Yf;
            end
          else
            begin
            LineFs:=0;
            LineLs:=0;
            Yf:=0;
            Yl:=0;
            YgapFrom:=StpAtPoints[yi,0];
            end;
          inc(StarSearch.StNsearch);
          end
        else
          begin
          if (StarSearch.StNsearch>1) and
             (StarSearch.StSearchDA[StarSearch.StNsearch-1].LineGapT=0) then
            begin
            StarSearch.StSearchDA[StarSearch.StNsearch-1].LineGapT:=
              StpAtPoints[yi,0]-StarSearch.StSearchDA[StarSearch.StNsearch].LineT;
            end;
          LineFs:=0;
          LineLs:=0;
          Yf:=0;
          Yl:=0;
          if StpAtPoints[yi,StpAtPointsMax+1]>PixWreq then
            begin
            YgapFrom:=StpAtPoints[yi,0];
            inc(StarSearch.BadLines, LineLs-LineFs);
            end;
          end;
      end;
    end;
  if StarSearch.StNfound>NofStMax then StarSearch.StNfound:=0;
  StLook:=StarSearch.StNfound;
  end;

function  SetIntBuffOfsTh: boolean;
  var
    Np, Ntr, Nl: word;
    ThS, ThE: word;
    ch: char;
  begin
  Ntr:=0;
  Dac3V:=DAC3VthStart;
```

```
ExposureTime:=0;
StarSearch.StNfound:=0;
ClearIandSandSR_PCSG(2);
repeat
  Np:=StoreFastSearch(ExposureTime);
  {$IfNDef REMOTE }
  ShowFastSearch(Np);
  {$EndIf }
  inc(Ntr);
  if Np>50 then
    begin
    if Ntr<5 then inc(DAC3V, 50) else inc(DAC3V,100);
    if Ntr>10 then LedTime1:=0;
    if Ntr>2 then ClearIandSandSR_PCSG(2);
    end;
  until (Ntr>15) or (Np<50);
if (Ntr>15) then SetIntBuffOfsTh:=false else SetIntBuffOfsTh:=true;
if (Ntr=1)  then
  begin
  ThS:=DAC3VthStart;
  LedTime1Srchset:=LedTime1;
  end;
if (Ntr>=2) then
  begin
  if Np>10 then ThS:=DAC3V+200 else ThS:=DAC3V+100;
  end;
ExposureTime:=ExpTSrchMax0;
Dac3V:=DAC3VthStart;
ClearIandSandSR_PCSG(2);
Ntr:=0;
repeat
  Np:=StoreFastSearch(ExposureTime);
  Nl:=StpAtPoints[0,0];
  {$IfNDef REMOTE }
  ShowFastSearch(Np);
  {$EndIf }
  inc(Ntr);
  if (Np<=200) or (Nl<=20) then
    begin
    if DAC3V>50 then dec(DAC3V, 50);
    if Ntr>8 then LedTime1:=2*LedTime1;
    end;
  until (Ntr>10) or (Np>200) or (Nl>20);
if (Ntr>=2) then ThE:=DAC3V else The:=DAC3VthStart;
if (ThE<ThS) and (Ntr>=2) then
  begin
  DAC3VthSet:=round(ThE+(ThS-ThE)*0.7)
  end
else DAC3VthSet:=ThS;
ExposureTime:=0;
ClearIandSandSR_PCSG(5);
Dac3V:=Dac3VthSet;
{
Np:=StoreFastSearch(ExposureTime);
ShowFastSearch(Np);
delay(1000);
}
ClearIandSandSR_PCSG(2);
end;
```

```
function  StSearch(Nstar:word): word;
  var
    Np: word;
    ch: char;
    i: word;
    UpLevel: boolean;
    ExpUp  : boolean;
    InitialClear   : boolean;
    Ntr    : word;
    StFound : word;
    StFoundB: word;
    StFoundBB: word;
    StFoundBBB: word;
    StFoundBBBB: word;
    ET    : word;
    ETB   : word;
    ETBB  : word;
    ETBBB : word;
    ETBBBB: word;
  begin
  UpLevel := true;
  ExpUp   := true;
  InitialClear := true;
  Ntr     := 0;
  StFound := 0;
  StFoundB:= 0;
  StFoundBB := 0;
  StFoundBBB:= 0;
  StFoundBBBB:= 0;
  ET      := 0;
  ETB   := 0;
  ETBB  := 0;
  ETBBB := 0;
  ETBBBB:= 0;
  LedTime1:=LedTime1Srch;
  SetIntBuffOfsth;
  StSearch:=0;
  ExposureTime:=ExpTsrchStart;
  for i:=0 to NofStmax do
    begin
    StarSearch.StSearchDA[i].ExpF:=0;
    StarSearch.StSearchDA[i].ExpO:=0;
    end;
  ClearIandSandSR_PCSG(2);
  repeat
    begin
    inc(Ntr);
    StFoundBBBB:=StFoundBBB;
    StFoundBBB:=StFoundBB;
    StFoundBB:=StFoundB;
    StFoundB:=StFound;
    EtBBBB:=EtBBB;
    EtBBB:=EtBB;
    EtBB:=EtB;
    EtB:=Et;
    {GenSynch;}
    Np:=StoreFastSearch(ExposureTime);
    ClearIandSandSR_PCSG(1);
    if (Np>=4) and (StpAtPoints[0,0]>1) then
      begin
```

```
      StFound:=StLook;
      ClearIandSandSR_PCSG(1);
      end
    else
      begin
      Starsearch.StNfound:=0;
      end;
    if StFound=Nstar then Et:=ExposureTime;
    {$IfNDef REMOTE }
    ShowFastSearch(Np);
    {$EndIf }
    ClearIandSandSR_PCSG(1);
    if Ntr>=2 then
      begin
      InitialClear:=false;
      end;
    if Np<5 then ExpUp:=true;
    if Np>=9999 then ExpUp:=false;
    if not InitialClear and (Np<5) and
         (ExposureTime>=ExposureTimeMax*0.75) then
      begin
      SetIntBuffOfsTh;
      end;
    if not InitialClear and (Np>=9999) and (ExposureTime<ExpTSrchMinO) then
      begin
      SetIntBuffOfsTh;
      end;
    if ExposureTime<=1               then ExpUp:=true;
    if ExposureTime>=ExposureTimeMax then ExpUp:=false;
    if not InitialClear then
      begin
      if ExpUp     then ExposureTime:=round(ExposureTime*1.5+1);
      if not ExpUp then ExposureTime:=trunc(ExposureTime*0.5);
      end;
    {if KeyPressed then ch:=ReadKey;}

    end;
    until (ch=#27) or (StFound{B}>=Nstar) or ( (StFound>0) and (Ntr>2) );
    {if (StFound=StFoundB) then}
     ExposureTime:=Et {B} {else ExposureTime:=EtB div 2;} ;

    StSearch:=Stfound;
  {halt;}
  end;


Begin
End.
```

```
Unit StAux01;   {StAux01.pas}

{Control auxilary devices coolers, housekiping multiplexers}
{ DAC multiplexer, power savig switches }

interface

uses
  Stdef01;

procedure LoadDAC12(V:word; DAC:byte);
procedure LoadDAC(V:word; DAC:byte);
procedure Anal_ON;
procedure Anal_OFF;
procedure SetMux2_3_0Ch(n:byte);
procedure LoadDtoA(N:byte);
procedure SwichDtoAMux(N:byte);
procedure Temp_OFF;
procedure Temp_ON;
procedure PCooler_ON;
procedure PCooler_OFF;
procedure VoltMonit(var Volt: TVoltages);

implementation

uses
  StAQ01, StPCSG01;


const
  WaitForDACstab = 25;
  WaitForMUXstab = 75;


procedure LoadDtoA(N:byte);
  var
    DtoA:byte;
  begin
  {
  Port[AuxDr]:= DtoA1_L or (N shl DACadrOfs) or Deselect;
  Port[AuxDr]:= (N shl DAcadrOfs) or Deselect or MuxEn1;
  }
  Port[AuxDr]:= DtoA1_L or (N and 7) or Deselect;
  Port[AuxDr]:= (N and 7) or Deselect or MuxEn1;
  end;

procedure SwichDtoAMux(N:byte);
  var
    DtoA:byte;
  begin
  Port[AuxDr]:= (N and 7) or Deselect or MuxEn1;
  end;

function BitTestW(W:word; Bit:byte):boolean;
  begin
    BitTestW :=  ( (W shr Bit) and 1 ) = 1;
  end;

procedure LoadDAC12(V:word; DAC:byte);
  var
```

```
    i:byte;
    j:word;
    k:byte;
  begin
  for i:=11 downto 0 do
    begin
    if not BitTestW(V, i) then
      begin
      Port[AuxDr]:=Deselect or SDin;
      Port[AuxDr]:=Deselect or Sclk or SDin;
      Port[AuxDr]:=Deselect;
      end
    else
      begin
      Port[AuxDr]:=Deselect;
      Port[AuxDr]:=Deselect or Sclk;
      Port[AuxDr]:=Deselect;
      end; {else}
    end;
  Port[AuxDr]:=Deselect;
  LoadDtoA(DAC);
  end;

procedure LoadDAC(V:word; DAC:byte); assembler;
  ASM
    cli
    mov  bx, V
    mov  cx, 12
    mov  dx, AuxDr
    mov  al, Deselect or SDin
    mov  ah, Deselect or Sclk or SDin
    mov  si, ax
    mov  ah, Deselect
    mov  al, Deselect or Sclk
    mov  di, ax
    out  dx, al
    {
    shl  bx, 16-12
    }
    shl  bx, 1
    shl  bx, 1
    shl  bx, 1
    shl  bx, 1
    @1:
    shl  bx,1
    jc  @2
    mov  ax, di
    out  dx, al
    xchg al, ah
    jmp @3
    @2:
    mov  ax, si
    out  dx, al
    xchg al, ah
    out  dx, al
    mov  ax, di
    xchg al, ah
    @3:
    out  dx, al
    loop @1
```

```
    mov  bl, DAC
    {shl  bl, DACadrOfs}
    mov  bh, DtoA1_L
    mov  ah, Deselect
    or   ah, bl         { ah := Deselect or DAC           }
    mov  al, ah
    or   al, bh         { al := Deselect or DAC or DtoA1_L }
    out  dx, al         { load DAC, select channel do NOT enable }
    mov  al, ah         { al := Deselect or DAC           }
    out  dx, al         { select channel                      }
    mov  cx, WaitForDACstab
    @4:
    loop @4
    or   al, MuxEn1
    out  dx, al
    mov  cx, WaitForMUXstab
    @5:
    loop @5
    sti
  end;

procedure SetMux2_3_0Ch(n:byte);
  begin
  Port[AuxDr]:=(Port[Auxdr] and not DACaddrEnMask) or (n and DACaddrMask);
  if n<8 then
    Port[LedDr]:=(Port[LedDr] and not Mux2_3EnMask) or MuxEn2
  else
    Port[LedDr]:=(Port[LedDr] and not Mux2_3EnMask) or MuxEn3;
  Port[AnalDr]:=(Port[AnalDr] and not Mux0addrMask) or MuxEn0 or Mux0ChHouse;
  end;

procedure Temp_ON;
  begin
  Port[AnalDr]:=Port[AnalDr] or Temp_En;
  end;

procedure Temp_OFF;
  begin
  Port[AnalDr]:=Port[AnalDr] and not Temp_En;
  end;

procedure PCooler_ON;
  begin
  Port[LedDr]:=Port[LedDr] or ExtOut2;
  end;

procedure PCooler_OFF;
  begin
  Port[LedDr]:=Port[LedDr] and not ExtOut2;
  end;

procedure Anal_ON;
  begin
  Port[AuxDr]:=Deselect or Anal_EN;
  Port[AnalDr]:=MuxEn0 or Mux0ChIntBuffOut;
  delay(2);
  LoadDAC(DAC3V,3);
  delay(1);
  LoadDAC(DAC1V,1);
  end;
```

```
procedure Anal_OFF;
  begin
  Port[AuxDr]:=0;
  end;

procedure VoltMonit(var Volt: TVoltages);
  var
    i: byte;
  begin
  Anal_ON;
  Temp_ON;
  {GenSynch;}
  for i:=0 to 15 do
    begin
    SetMux2_3_0Ch(i);
    delay(1);
    Volt[i]:= GetADCwait_PCSG;
    end;
  for i:=16 to 23 do
    begin
    SetMux0Ch(i-16);
    delay(1);
    Volt[i]:= GetADCwait_PCSG;
    end;
  end;

Begin

End.
```

```
unit Streml;

{ Used for serial communication - interface to CCD.ASM kernel }

interface

{$IfDef REMOTE}

function PutChar(ch: char):boolean;
function PutCharW(ch: char):boolean;
function PutStr(strg:string):boolean;
function PutStrW(strg:string):boolean;
function GetChar:char;
function GetCharW:char;
function GetBytesW(n:byte):string;
function GetToRetW:string;

{$EndIf}


implementation

uses
  StDef01, DOS;

procedure MChdel;
  var
    j: longInt;
  begin
  for j:=0 to 10000 do
    begin
    end;
  end;

{$IfDef REMOTE}

function PutChar(ch: char):boolean;
  var
    Reg: Registers;
  begin
  Reg.al:=byte(ch);
  Reg.ah:=$FF;
  Intr($11,Reg);
  PutChar:=Reg.ah=0;
  {MCHdel;}
  end;

function PutCharW(ch: char):boolean;
  var
    Reg: Registers;
  begin
  Reg.al:=byte(ch);
  Reg.ah:=$0;
  Intr($11,Reg);
  PutCharW:=Reg.ah=0;
  {MChdel;}
  end;

function PutStr(strg:string):boolean;
  var
```

```
    Reg: Registers;
    n, i: byte;
    OK:boolean;
  begin
  OK:=false;
  n:=byte(strg[0]);
  for i:=1 to n do
    begin
    PutChar(strg[i]);
    end;
  PutStr:=true;
  end;

function PutStrW(strg:string):boolean;
  var
    Reg: Registers;
    n, i: byte;
    OK:boolean;
  begin
  OK:=false;
  n:=byte(strg[0]);
  for i:=1 to n do
    begin
    PutCharW(strg[i]);
    end;
  PutStrW:=true;
  end;

function GetChar:char;
  var
    Reg: Registers;
  begin
  Intr($10,reg);
  if Reg.ah=0 then GetChar:=char(Reg.al)
            else GetChar:=#0;
  end;

function GetCharW:char;
  var
    Reg: Registers;
  begin
  repeat
    begin
    Intr($10,Reg);
    getCharW:=char(Reg.al);
    end;
    until Reg.ah=0;
  end;

function GetBytesW(n:byte):string;
  var
    Reg: Registers;
    St:string;
    nn:byte;
  begin
  St:='';
  nn:=0;
  repeat
    begin
    St:=St+GetCharW;
```

```
    inc(nn);
    end;
    until nn>=n;
  GetBytesW:=St;
  end;

function GetToRetW:string;
  var
    Reg: Registers;
    St:string;
    nn:byte;
    ch: char;
  begin
  St:='';
  nn:=0;
  repeat
    begin
    Ch:=GetCharW;
    St:=St+Ch;
    inc(nn);
    end;
    until (Ch=#13) or (nn>=255);
  GetToRetW:=St;
  end;

{$EndIf}

BEGIN
END.
```

```
; ******************************************************************
; *                                                                *
; *     Turbo Pascal Runtime Library Version 6.0                   *
; *     Main module                                                *
; *                                                                *
; *     Copyright (C) 1988, 89 Borland International               *
; *     Modifications Copyright (C) 1989 Paradigm Systems          *
; *                                                                *
; *     This module must be inserted into TURBO.TPL using the      *
; *     Turbo Pascal 6.0 run-time library source kit.              *
; *                                                                *
; *     Modified by M. Chmielowski to operate with TP 6.0          *
; *     Tested on remote system 22ed Sep. 1992                     *
; *     Replace MAIN.ASM in c:\Tp\RTL\SYS                          *
; *     Delete *.OBJ in \SYS and recompile                         *
; *                                                                *
; ******************************************************************

        TITLE   MAIN

        INCLUDE SE.ASM

DATA    SEGMENT WORD PUBLIC

; Externals
        EXTRN   OvrHeapSize:WORD,OvrHeapOrg:WORD,OvrHeapPtr:WORD
        EXTRN   OvrHeapEnd:WORD,OvrLoadList:WORD,HeapOrg:DWORD
        EXTRN   HeapPtr:DWORD,HeapEnd:DWORD,FreeList:DWORD
        EXTRN   HeapError:DWORD,ExitProc:DWORD,ExitCode:WORD,
        EXTRN   ErrorAddr:DWORD,PrefixSeg:WORD,InOutRes:WORD
        EXTRN   Input:BYTE,Output:BYTE,SaveInt00:DWORD

DATA    ENDS

CODE    SEGMENT BYTE PUBLIC

        ASSUME  CS:CODE, DS:DATA

; Externals
        EXTRN   AssignText:NEAR, ResetText:NEAR, RewriteText:NEAR
        EXTRN   CloseText:NEAR

; Publics
        PUBLIC  InitTurbo, HaltTurbo, HaltError, PrintString
        PUBLIC  Terminate

;
;       Initialize the runtime library. First instruction in any program
;       is a call to this routine.
;
InitTurbo:
        MOV     DX, SEG DATA            ; Initialize DS
        MOV     DS, DX

        XOR     BP, BP                  ; End of stack frame chain
```

```
        MOV     AX, SP                  ; Compute first free segment
        ADD     AX, 4+15                ; address in AX
        MOV     CL, 4
        SHR     AX, CL
        MOV     DX, SS
        ADD     AX, DX

        MOV     HeapOrg.seg,AX          ;Initialize heap manager
        MOV     HeapPtr.seg,AX          ;variables
        MOV     FreeList.seg,AX
        MOV     DX,ES:1000h             ; pspMemTop
        MOV     HeapEnd.seg,DX
        MOV     HeapError.ofs,OFFSET CS:HeapFailure
        MOV     HeapError.seg,CS

        PUSH    DS                      ; Install interrupt handlers
        PUSH    CS
        POP     DS
        MOV     DX, OFFSET Int00Handler
        xor     bx, bx
        mov     es, bx
        mov     es:[bx], dx
        mov     es:[bx+2], ds
        pop     ds                      ; Restore DS

        RETF                            ; Back to main program

; Default heap error handler. Return 0 to indicate run-time error.

HeapFailure:

        XOR     AX,AX
        RETF    2


;
; Divide by zero interrupt handler. Control arrives here upon
; executing a DIV or IDIV instruction with a zero divisor.
;
Int00Handler:
        MOV     AX, 200

;
; RunError standard procedure
;
HaltError:
        POP     CX
        POP     BX
        JMP     SHORT Terminate

;
; Halt standard procedure
;
HaltTurbo:
```

```
        XOR     CX, CX
        XOR     BX, BX

; Terminate program and restart the application
; In    AX    = Exit code
;       BX:CX = Error address (or NIL)

Terminate:
;       jmp     InitTurbo               ; Simply restart

;
;       This code is for use with TDREM
;
        xor     bx, bx                  ; Make the interrupt vector table addressable
        mov     es, bx

        pushf                           ; Push the PSW
        push    bx                      ; IP
        push    bx                      ; CS
        jmp     dword ptr es:[000ch]    ; Call the break handler



PrintString:
@@1:    MOV     AL, CS:[BX]
        OR      AL, AL
        JE      @@2
        CALL    PrintChar
        INC     BX
        JMP     SHORT @@1
@@2:    RET

; Print byte in decimal
; In    AL = Value

PrintDec:
        MOV     CL, 100
        CALL    @@1
        MOV     CL, 10
        CALL    @@1
        JMP     SHORT @@2
@@1:    XOR     AH, AH
        DIV     CL
@@2:    ADD     AL, '0'
        PUSH    AX
        CALL    PrintChar
        POP     AX
        MOV     AL, AH
        RET

; Print word in hex
; In    AX = Value

PrintHex:
        PUSH    AX
```

```
        MOV     AL, AH
        CALL    aa1
        POP     AX
aa1:    PUSH    AX
        MOV     CL, 4
        SHR     AL, CL
        CALL    aa2
        POP     AX
        AND     AL, 0FH
aa2:    ADD     AL, '0'
        CMP     AL, '0'+10
        JB      PrintChar
        ADD     AL, 'A'-'0'-10

; Print character
; In    AL = Character

PrintChar:
        RET

; Empty string
ZeroString      DB      0

CODE    ENDS

        END
```

CCD ROM software
T. Nolam / M. Chmielowski
Revision Histor

5/11/92	v1.00	Preliminary Release
8/1/92	v1.01	Added program loader, reorganized EEPROM
8/19/92	v1.02	Program control commands, startup opts.
9/30/92	v2.00	Modyfied

Introduction

The ROM software is in two parts: the ROM Monitor and the Turbo Debugger Remote Kernel. These two parts are built from separate sources, compiled and linked into separate binary images, and placed at known locations in the EEPROM memory on the CCD CPU board. The memory map is as follows:

F0000 - FDFFF	Unused
FE000 - FF2FF	Turbo Debugger Remote Kernel
FF300 - FFFEF	ROM Monitor
FFFF0 - FFFF5	BOOT
FFFF6 - FFFF7	KERNEL Protection TEST point

The ROM Monitor Program

The ROM Monitor is a custom-written startup and diagnostic program. It gains control when the CPU is reset or when the power is applied. It runs the following tests on the CPU and peripheral devices:

0. RAM pattern test - 55aa written to all locations and read back
1. RAM pattern test - aa55 written and read back
2. Byte access test - single bytes written separately to high and low order bytes of RAM
3. Address test - each RAM location written with its own address COMPIMENT and read back
4. 8259 Interrupt controller test - addressability and response checked
5. 8255 PPI test - addressability and response checked
6. 8254 PIT test - set up three counters and verify that they are counting
7. 8251 UART test - programmed for 9600 baud, 8 data bits, no parity, 1 stop bit (requires baud rate generator output from 8254). Character transmitted and error flags checked
8. EEPROM access test - data read inver writte then restore to:
   a) $FFFF6 - hardware protected Kernel - should be *DENIED
   b) $F0000 - EEPROM1 test

c) $D0000 - EEPROM2 test

These tests take about 6 seconds with a 4MHz clock. The ROM monitor then prints the status of the tests. Output is done through the serial port, so if a terminal (or PC terminal emulator) is connected the results will be visible. The ROM monitor enters a command loop, in which it reads and processes commands from the terminal.

The ROM monitor understands a number of commands.
The commands and their syntax are as follows:

Wssss:oooo	write startup address ssss:oooo
G[ssss:oooo]	go to startup address
P	print diagnostics
Dnnnn [mm]	dump from segment nnnn mm segments [8]
Lnnnn mm	load into segment nnnn, mm bytes
(binary data follows)	
R	reset from location F000:FFF0
S	'reset' from location D000:FFF0
9	baud rate 9600 Hz
2	baud rate 19200 Hz
4	baud rate 38400 Hz
1	baud rate 115k Hz

All numbers are in hex.

The 'W' command is used to place the address (segment and offset) of the user program entry point into EEPROM. This address is used by the ROM monitor to transfer directly to the user program when the board is not connected to a debug monitor. The actual ROM address where this 2-word value is stored is FDFFCh (offset first, then segment, Lo byte then Hi byte).
This is the highes 4 bytes just before hardware protected area.
e.g. W0102:0304 gives:

```
$FDFFC = $04
$FDFFD = $03
$FDFFE = $02
$FDFFF = $01
```

The 'G' command transfers control to a user-specified entry point, or to the EEPROM transfer address written by the 'W' command if no address is specified. The 'G' command can be used with the program loader, by loading the program and then transferring to its entry point.

The 'P' command prints the diagnostics as though the ROM monitor had just started up.

The 'D' command is the data dumper.

The 'L' command is the program loader interface.

The 'L' command is intended for use by the "pload" program loader. After the command is typed, binary data is accepted by the serial port and put into memory at the specified segment address. A maximum of 128 (80h) bytes may be loaded at a time. This is due to the EEPROM write limitation. Larger quantities of data are loaded in separate blocks. All addresses given to these commands and displayed by the dump command are segment addresses, also known as paragraph addresses. To convert a segment address to a full 20-bit address, multiply by 16 (10h). For example, the command "D200" causes memory starting at segment 200h (absolute address 2000h) to be displayed. The command "Lf000 80" causes the following 80 bytes of binary data to be loaded into EEPROM at absolute address F0000h, which is the start of EEPROM. Note that in the command syntax there must not be a space between the command character and the segment address. The program loader checks to make sure that no data is ever written to addresses FE000-FFFF5 (the kernel and tdrem area).

Whil in the command loop waiting for a user command, the ROM monitor checks for a NUL character on the input port. This is the handshake character issued by the Turbo Debugger in remote mode. If this character is received, the ROM monitor transfers control to the Turbo Debugger Remote Kernel.

The Turbo Debugger Remote Kernel

The remote debugging kernel is a semi-custom software module supplied by Paradigm Systems and modified for the CCD board installation. It communicates with the Turbo Debugger supplied with Borland C and Pascal languages to allow a program to be debugged remotely on the target CPU board. For this operation, a PC is connected through its COM port to the CPU board. The Turbo Debugger is brought up, using a command line like

```
td -r -rs1 <progname>
```

where "-r" is the remote debugging flag, "-rs1" specifies 9600 baud, and "<progname>" is the name of a program to be loaded and executed on the remote CPU. The program is loaded from the PC disk, transferred to the CPU by the debugger, and if all goes well the debugger shows the main program screen just as if the program were being debugged locally. All of the normal Turbo Debugger features are available (but it is kind of slow).

The program must have been built specially for the remote CPU. The requirements are as follows:

1. Link in a non-standard startup assembly language file in place of the startup code in the system library. Create an exe file (but name it .ROM instead of .EXE), and a segment map file (.MAP).

2. Run the Paradigm Systems "locate" program to create

an absolute image from the .ROM output (this file is now named .EXE). In the locate directives, specify RAM and simulated ROM within the RAM area of the CPU board. The loader can not load into EEPROM, nor can breakpoints be set in the debugging process on code in EEPOM. Thus the program code and data must fit into 64KB, the size of RAM on the CPU board.

3. Don't use any MS-DOS or ROM-BIOS calls, or call any system or library function that uses any of these.

Building a Program for Loading into EEPROM

If a program is to be loaded into EEPROM, it must be relocated in a different way than if it is to be loaded under control of TDREM. In the former case, the program's "ROM" is actually within the target board's RAM area. This is because TDREM wants to load and run a program just as if it were working with MSDOS. TDREM loads the program code and data into RAM, where it can set and clear breakpoints at will. In the latter case, the program's ROM is located in EEPROM, where it belongs. The locate directives are used to make a binary image, with the code segment located at F000, and the data segment located at 100 (which is above any kernel RAM).

The program's initialized data is going to be a bit of a problem. It must reside originally in EEPROM, then be copied to RAM by the startup code. This means that a different Pascal startup routine must be used for EEPROM code than was used with TDREM. The EEPROM startup routine copies initialized data to RAM, then transfers control to the Pascal main program in EEPROM. This is the normal approach for ROM-based programming.

After the program is loaded into EEPROM, you can write its entry point into EEPROM so the ROM monitor can transfer directly to the program. This provides autonomous startup of the board, without user intervention. The next section describes the startup operations.

Startup Control

The ROM monitor goes through the following sequence in deciding whether to give control to the user program.

1. If a terminal is connected to the RS-232 port, and drives the board's DSR input (pin 20 on the 25-pin connector), the program invokes the ROM monitor. Note that pin 20 is the normal DTR output pin from the computer. The cable has a null-modem wiring to bring this into the DSR input.

2. If the terminal is connected but does not drive DTR, the ROM

monitor waits approximately one second after it completes its memory diagnostics. If it receives an 'Esc' character on the serial port during this time, the program invokes the ROM monitor. The program is tolerant of extra 'Esc' chars, so in practice, it is fairly easy to tap this key until the ROM monitor messages come up.

3. If neither of these conditions is in effect, i.e. the DTR is inactive and the serial port does not receive an 'Esc' char, then the ROM monitor does not print its diagnostics, but instead transfers directly to the user program entry point, which is stored in the EEPROM at address FFFF6h.

ROM Monitor Routines Available to User Programs

If TDREM is not used, the kernel routines for serial port I/O can be used by the Pascal program. The descriptions are as follows:

INT 10h - Get character
    Returns character in al, zero in ah.
    Returns ax = FFFF if no character present.

INT 11h - Put character
    Call with ASCII char in al, return flag in ah.
    Returns: if ah is zero, waits until character is successfully queued before returning with ah still zero. If ah is non-zero, returns immediately with ah zero if the character was queued, ah non-zero if the queue was full and the operation needs to be retried.

The port I/O software interrupts use queued input and output. The queues are 256 characters deep. Arriving characters will be buffered until 256 have been received; additional characters will be lost. Calls to INT 10h will pick up characters from the queue and make room for new incoming characters. When the queue is empty, the function returns immediately with an empty indication.

Likewise, up to 256 characters can be placed in the output queue by int 11h. However, if the output queue is full when INT 11h is called, it will not return to the caller until space is available and the character is queued successfully.

The External Program Loader

An MS-DOS utility program PPLOAD.EXE is provided in this release.
Source in PPLoad.pas.
Its purpose is to transfer a binary image in an MS-DOS disk file over the serial port into the EEPROM under control of the ROM monitor. The syntax is:

```
pload binfile seg
```

where "binfile" is the name of the binary data file to be loaded, "seg" is the segment address where the program is to be loaded on the target CPU board using port COM 2. Typically you will use a segment address of F000, which is the start of EEPROM.

To use the program loader, first create the binary image. PLOAD does not understand intel hex (which TDREM uses). Instead, use the "hexfile binary" directive to Paradigm Locate to create a binary file in the exact image of EEPROM. The file size must be 56K or smaller, since that's all the room there is in EEPROM. Next, connect the CPU board to the PC's serial port, start up the terminal emulator of your choice, and boot up the ROM monitor by applying power or resetting the board. You will see the diagnostics appear on the screen. Next, exit the terminal emulator and run PLOAD with the appropriate command line arguments. As a final step, you may want to bring up the terminal emulator again and verify that the program has loaded correctly by dumping some part of memory.

The program loader is protected from writing to kernel EEPROM. However it is not protected from writing to kernel RAM (segment < 100). Doing so will cause the ROM monitor to crash. Various error messages are possible from the program loader. The best way to figure out what they mean is to consult the source code files PPLOAD.pas and CCD.ASM.

Unsolved Problems

Support for queued communication (INT 10h and INT 11h) do NOT operate correctly for more then 300 characters.

Loading Boot jump, Boot program, and TDremote Kernel

Load each file separately into EEPROM filled with $FF

At $FDFFC write start address of user program.
ofset then segment
bytes are placed in order Lo byte first then Hi byte
Command: W0102:0304 gives

```
        in Lo   $6FFE = $04
                $6FFF = $02
        in Hi   $6FFE = $03
                $6FFF = $01
```

For soft RESET store seg: $FFFF, ofs: $0000

```
        in Lo $6FFE = $00
              $6FFF = $FF
        in HI $6FFE = $00
              $6FFF = $FF
```

For start address equal jump to reset enter 00FF at 6FFC
in each EEPROM

Check Sums are without start address

Loading CCD.EXE image into EEPROM
into memory $FF300

CCD.HX0 - for EEPROM1 Lo (bits 0-7)
CCD.HX1 - for EEPROM1 Hi (bits 8-15)

file ofset	$0
file length	$A38
device ofset	$7980

Check Sum:

EEPROM Lo	$8FF6
EEPROM Hi	$7C1B

Then load TDREM.HX0 and TDREM.HX1
into memory $FE000

file ofset	$0
file length	$12FF
device ofset	$7000

Check Sum for both CCD and TDREM:

EEPROM Lo	$0301
EEPROM Hi	$EE48

with reset address

EEPROM Lo	$0202
EEPROM Hi	$ED49

Saved as KER4L.HEX and KER4H.HEX

```
//    Define the target system memory map

;             Memory configuration - physical
;
;       $00000 - $0FFFF 64k bytes of RAM1
;       $10000 - $1FFFF 64k reserved for GHOST image of 64k RAM1
;       $20000 - $2FFFF 64k bytes of RAM2
;       $30000 - $3FFFF 64k reserved for GHOST image of 64k RAM2
;       $40000 - $BFFFF     reserved - NO physical MEM
;       $C0000 - $CFFFF 64k reserved for GHOST image of 64k EEPROM2
;       $D0000 - $DFFFF 64k bytes of EEPROM2
;       $E0000 - $EFFFF 64k reserved for GHOST image of 64k EEPROM1
;       $F0000 - $FFFFF 64k bytes of EEPROM1
;
;             Memory configuration - software
;
;       $00000 - $00FFF  4k RAM kernel
;       $01000 - $0FFFF 60k RAM to use
;       $10000 - $1FFFF 64k DO NOT use
;       $20000 - $2FFFF 64k RAM to use
;       $30000 - $DFFFF     DO NOT use
;       $C0000 - $CFFFF 64k EEPROM2
;       $E0000 - $EFFFF 64k DO NOT use
;       $F0000 - $FDFFF 56k EEPROM1 to use
;       $FE000 - $FF2FF 7+k EEPROM1 reserved for T Debugger kernel - may be used
;                                              in final version
;       $FF300 - $FFFFF 1-k EEPROM reserved for ROM kernel

map     0x00000 to 0x00fff as rdonly    // RAM kernal  4k
map     0x01000 to 0x0FFFF as rdwr      // RAM to use 60k
map     0x10000 to 0x1FFFF as reserved  // ghost of RAM1
map     0x20000 to 0x2FFFF as rdwr      // RAM2 to use 64k
map     0x30000 to 0x3FFFF as reserved  // ghost of RAM2
map     0x40000 to 0xBFFFF as reserved  // NO physical MEM
map     0xC0000 to 0xCFFFF as reserved  // ghost of EEPROM2
map     0xD0000 to 0xDFFFF as rdonly    // EEPROM2 64k
map     0xE0000 to 0xEFFFF as reserved  // ghost of EEPROM1
map     0xF0000 to 0xFDFFF as rdonly    // EEPROM to use 56k
map     0xFE000 to 0xFF2FF as rdonly    // EEPROM T debugger kernel 7+k may be use in future
map     0xFF300 to 0xFFFFF as rdonly    // EEPROM ROM kernel 1-k

cputype I8086

display all
//absfile       axe86 filename=st5tst.exe format=TD20   // File for TD
hexfile binary filename=st5Dseg.bin offset=0xD0000 size=64
hexfile binary filename=st5Fseg.bin offset=0xF0000 size=64
listfile segments regions                        // Output a segment map
initcode stack                          // TP requires stack initialization
dup     DATA DATAR
class   ??LOCATE = 0XF000                        // Assign code address
class   DATA = 0X2000                   // Assign data address
order   ??LOCATE CODE               // Place in order into ROM1
class   STACK = 0X100
order   DATA HEAP
class   DATAR = 0XD000                   // Assign data in ROM address
output  CODE ??LOCATE DATAR             // Classes in the output file
```

STAR TRACKER BOARD

I/O and Interrupts Map

I/O Ports

10x0	r/w	stop the clk
11x0	r/w	8259 Interrupt controller port
11x2	r/w	8259 Interrupt controller port
12x0	r/w	8255 Peripheral Port Interface A
12x2	r/w	8255 Peripheral Port Interface B
12x4	r/w	8255 Peripheral Port Interface C
12x6	r/w	8255 Peripheral Port Interface W (control)
13x0	r/w	8251 USART controller port
13x2	r/w	8251 USART controller port
14xx	r	a/d inputs from ccd board
15x0	r/w	8254 Programmable Timer controller port
15x2	r/w	8254 Programmable Timer controller port
15x4	r/w	8254 Programmable Timer controller port
15x6	r/w	8254 Programmable Timer controller port
16xx	r/w	Watchdog Timer Reset Port and EEPROM write (one byte) enable write OfHh to this port to enable writes to eerpom write any other word to disable writes to eeprom
17xx	r/w	Waveform generator

Interrupts

INT 0	USART Data Rdy interrupt
INT 1	USART Tx Rdy interrupt
INT 2	Wavegen Interrupt
INT 3	Programmable Time Base Interrupt
INT 4	Programmable Timer Interrupt
INT 6	Wavegen Timer Interrupt

```
# make file for Pload.C
pload.exe: pload.c
        TCC -I$c:\TC\include Pload
```

```
rem Batch program for building executable Star Tracker program
rem targeted for remote processor.

call makebat.bat st5.exe
```

C:\TP\RTL\REMBIN\MAKEBAT.BAT 2/8/93

```
rem Batch program for building ARC Star Tracker program
rem loading into remote target and for loading assembler
rem kernel (in two copies) and setting kernal start addres
rem for automatic execution of Star Tracker program
del st5tst.exe
del %1
make -B %1
rem td -rp2 -rs3 -l st5tst
ppload1.exe St5Dseg.bin $D000
ppload1.exe c:\tomnew\ccdd\ccddseg.bin $DF30
ppload1.exe St5Fseg.bin $F000 wF000:0000
turbo
```

```
#
#  MAKE file for building the ARC Star Tracker program for use with TDREM
#  or for execution from ROM.
#

TDREM           =       0                              # 0 - ROM build, 1 - T

TPOPTS   =      /Tc:\tp\rtl\bin /B # look for TURBO.TPL in remote version
# use TPC.CFG
# TPOPTS        =       $(TPOPTS) /DREMOTE
# /DREMS
TPOPTS   =       $(TPOPTS) /$M$4000,$2000,$2000
!if     $(TDREM) == 0
TPOPTS   =       $(TPOPTS) /DREMS
LOCOPTS =        -cROM.CFG -Ee
CFG              =       rm
!else
TPOPTS   =       $(TPOPTS) /V /DTDREM
LOCOPTS =        -Aa
CFG              =       td
!endif

LOCOPTS =        $(LOCOPTS)


.pas.exe:
        tpc $*.pas $(TPOPTS)
        locate $(LOCOPTS) $*.exe
```

```
/* Turbo Pascal config file for cmpilation for REMOT target */

/Tc:\tp\rtl\bin
/DREMOTE
/GP /B /$N- /$E- /$R+ /$G- /$D+ /$S+ /$V+ /$X+ /$L+ /$A+
/Uc:\tp\rtl\bin;c:\tp\rtl\bin\tpu
```

WIRING LIST

Boards List

CCD Internal Connection List

External Devices List

Board to Board Wiring List

Wiring List

Terminology used in this document

Boards and external devices are listed and numbered starting from CCD

External Devices (ED)	- CCD/Peltie Coolers/Thermistors/LEDs
Board 1	- Amplifiers/Drivers
Board 2	- ADC/DAC
Board 3	- Power supply
Board 4	- PCSG (Programmable Clock Signal Generator)
Board 5	- CPU
Board 6	- Memory
Board 7	- RS 422
Host Computer Connector (HC)	- In Test Unit use Serial Coonector and Power Connector (see end of this document)

Notation

B# - Board number, i.e. B2-JP63(4)- DGND, indicates board 2, JP63, pin 4 marked on silkscreen as DGND

JP# or J# - is used to designate a physical location of a wire connection on the silkscreen and the schematic.

JP#(#) or J#(#)- is used to designate a wire cluster, where cluster # is marked on the silkscreen and the schematic, number in brackets indicates wire position in cluster (pin 1 is marked with square).

Signal name - as marked on Board 1 drawings, for remaing boards signal name is given only for redundancy.

Dots e.g. (#a..#b) or (#a)

.

.

(#b)

means range of wires from #a to #b.

N.C. - Not Connected

Names are not case sensitive

There is ERRATUM for Board 2, see end of board 2 description.

Engineering Unit :

Boards 1, 2, and 3 and interconnected as stated in this list. Boards 4, 5, 6, and 7 are interconnected as stated in this list. Interconnection between the first three boards to 4, 5, 6, and 7 are replaced by test connecters CON0, CON1, CON3 (no CON2). Subsituted connections are defined in the list as TEST UNIT connections. Host Computer connector (HC) is replaced by two test connections, serial communication CONS and power supply connector CONP.

****

CCD internal connections

****

CCD signal	CCD pin #	CCD signal	CCD pin #
SUB	11	SUB	12
CDB	10	TDB	21
IAG	3	IAG	19
ABG	2	ABG	20

Total: 4 wires (8 pins) are connected with wires soldered to CCD pins.

****

CCD Not Connected

****

CCD signal	CCD pin #	
OUT 2	7	N.C.
OUT 3	6	N.C.

Total: 2 pins are N.C.

****

CCD - Board 1

16 wires listed for Board 1 interconnections

****

Total: 22 pins
4 wires internal connections on CCD
2 CCD pins are Not Connected
16 wires to B1

External devices

****

Peltie Coolers

NOT in TEST UNIT

2 wires listed for Board 1 interconnections

****

Thermistors:
Lens external
CCD temperature
Peltie Cooler Hot

6 wires listed for Board 1 interconnections

****

LED external

8 wires listed for Board 3 interconnections

****

Total: 16 wires
10 wires to devices outside (LEDs and Lens thermistor)
6 to devices inside (Peltie Cooler, CCD and Hot thermistors)

****

BOARD 1

****

CCD - Board 1

(CCD interface section)

CCD signal	CCD pin #	Board 1 signal
AMP GND	9	CCD_AMP_GND
OUT 1	8	CCD_OUT1
ADB	5	CCD_ADB
SUB	1	CCD_SUB1
SUB	22	CCD_SUB2
SUB	12	CCD_SUB3
TDB	21	CCD_TDB
IAG	19	CCD_IAG
ABG	2	CCD_ABG
SAG1	4	CCD_SAG1
SAG2	17	CCD_SAG2
TMG	18	CCD_TMG
TRG	13	CCD_TRG
SRG1	14	CCD_SRG1
SRG2	15	CCD_SRG2
SRG3	16	CCD_SRG3

Total: 16 wires from B1; 22 pins of CCD - 2 are N.C., 4 are connected internally on CCD.

****

Cooler/thermistors - Board 1 <u>NOT in TEST UNIT</u>

PC (top)	Red wire (+)	TE Cooler 1
PC (bottom)	Black wire (-)	TE Cooler 2

Internal thermistors - Board 1

Therm. CCD 1	TEMP_CCD
Therm. CCD 2	TEMP_GND
THERM. Cooler Hot 1	TEMP_PCB
THERM. Cooler Hot 2	TEMP_GND2

External thermistor - Board 1

THERM. LENS 1	TEMP_LENS
THERM. LENS 2	TEMP_GND3

(polarity of thermistors is not specified)

Total: 8 wires from B1; 2 wires are for external thermistor; polarity of PC is specified.

****

Board 1 - Board 2

Board 1 signal	Board 2 signal	Conn. #
TEMP_OUT	TEMP_OUT	B2-JP66
A3	MUX_ENO_G	B2-JP52
AMP1_OUT	AMP1_OUT	B2-JP18(1)
INT_BUFF_OFS	INT_BUFF_OFS	B2-JP29
MUX_OUT	MUX_OUT	B2-JP17
Test_Inp1	Test_Inp1	B2-JP39
UmIAG	UmIAG	B2-JP26(1)
UmABG	UmABG	B2-JP26(2)
U-ABG	U-ABG	B2-JP27
SRG1C	SRG1C	B2-JP24(2)
SRG2C	SRG2C	B2-JP24(3)
SRG3C	SRG3C	B2-JP24(4)
ABGC	ABGC	B2-JP24(1)
ABGI	ABGI_out	B2-JP34(1)

Total: 14 wires B1 to B2

****

Board 1 - Board 3

Board 1 signal	Board 3 signal	Conn. #
PC+I	PC+I	B3-JP50(2)
PcGND	PcGND	B3-JP50(1)
REF+5Vt	REF+5t_sw	B3-JP80
MUX_VM_OUT	MUX_VM_Out	B3-JP46
U+12c	U+12c	B3-JP21(5)
U+12a	U+12a	B3-JP21(4)
AGND	AGND	B3-JP21(3)
U-12a	U-12a	B3-JP21(2)
DGND	DGND	B3-JP21(1)
U+12cDr	U+12c	B3-JP43(6)
U+6Dr	U+6Dr	B3-JP43(5)
U+2Dr	U+2Dr	B3-JP43(4)
U-11Dr	U-11Dr	B3-JP43(2)
U+5Dr	U+5Dr	B3-JP43(3)
DDGND	DDGND	B3-JP43(1)
Bias+M	Bias+M	B3-JP44(2)
Bias-M	Bias-M	B3-JP44(1)
Bias22mon	Temp_OUT	B3-JP71

Total: 18 wires B1 to B3

****

Board 1 - Board 4 --

Board 1 signal	Board 4 signal	Conn. #	TEST UNIT
/INT_POL	INT_POL_POS	B4-J1(1)	CON3(40)
INT_SAMPLE	INT_SAMPLE	B4-J1(2)	CON3(39)
INT_RESET	INT_RESET	B4-J1(3)	CON3(38)
IAGC	IAG	B4-J31(9)	CON3(32)
SAG1C	SAG1	B4-J1(11)	CON3(30)
SAG2C	SAG2	B4-J1(12)	CON3(29)
TMGC	TMG	B4-J1(10)	CON3(31)
TRGC	TRG	B4-J1(13)	CON3(28)

Total: 8 wires B1 to B4

****

Board 1 - Board 5

0 wires

****

Board 1 - Board 6

0 wires

****

Board 1 - Board 7

Board 1 signal	Boards 7 signal	Conn. #	TEST UNIT
A0	HK2 (MUXADR0)	B7-JP32(1)	CON1(38)
A1	HK2 (MUXARD1)	B7-JP32(2)	CON1(18)
A2	HK2 (MUXADR2)	B7-JP32(3)	CON1(37)
IAGI	HK2 (IAGI)	B7-JP32(4)	CON1(36)

Total: 4 wires B1 to B7

****

Board 1 - HC

0 wires

****

```
************************************************

Board 1 -  N.C

Board 1 signal

DGND2
AGND2

DDGND2
Bias-

Total: 4 wires on B1 are N.C.

*******************************************
                                                      Board 1

Total: 68 wires from B1;
                 16 to CCD
                 8  to PC/therm.
                 14 to B2
                 18 to B3
                 8  to B4
                 0  to B5
                 0  to B6
                 4  to B7
                 0  to HC
                 *********
                 4 are N.C.
*******************************************
```

****

BOARD 2

****

Board 2 - External Devices

0 wires

****

Board 2 - Board 1

14 wires listed for Board 1 interconnection

****

Board 2 - Board 2 internal jumpers

Board 2 signal	Conn.#	Board 2 signal	Conn. #
INT_SAMPLE	B2-JP87	Grounging	B2-JP100
NAND_Spare_OUT	B2-JP98	INT_/RESET	B2-JP94
INT_RESET	B2-JP95	ABGCc2	B2-JP57

Total: 6 signals are connected with jumpers

****

Board 2 - Board 3

Board 2 signal	Conn.#	Board 3 signal	Conn. #
U+12c	B2-JP63(8)-V+c	U+12c	B3-JP75(8)
AGND	B2-JP63(7)-AGND	AGND	B3-JP75(7)
U-12c	B2-JP63(6)-V-c	U-12c	B3-JP75(6)
U+5c	B2-JP63(5)-Vcc	U+5c	B3-JP75(5)
DGND	B2-JP63(4)-DGND	DGND	B3-JP75(4)
U+12a	B2-JP63(3)-V+	U+12a	B3-JP75(3)
U+5Da	B2-JP63(2)-VccA	U+5Da	B3-JP75(2)
U-12a	B2-JP63(1)-V-	U-12a	
DAC_STR_A	B2-JP64(3)	DAC_STR_A	B3-JP72(3)
DAC_STR_B	B2-JP64(2)	DAC_STR_B	B3-JP72(2)
DAC_STR_C	B2-JP64(1)	DAC_STR_C	B3-JP72(1)
INH_MUX2	B2-JP76(1)	INH_MUX2	B3-JP78(2)
INH_MUX3	B2-JP76(2)	INH_MUX3	B3-JP78(1)
INT_BUFF_OFS_M		INT_BUFF_OFS	
	B2-JP67		B3-JP74(4)
UmIAG_M	B2-JP69	UmIAG_M	B3-JP74(2)
UmABG_M	B2-JP70	UmABG_M	B3-JP74(3)
U-ABG_M	B2-JP68	U-ABG_M	B3-JP74(1)
REF+5t	B2-JP25	REF+5t	B3-JP79
TEMP_EN_G	B2-JP77	TEMP_EN_G	B3-JP81
ANAL_EN	B2-JP65	ANAL_EN	B3-JP73

Total: 20 wires B2 to B3

****

Board 2 - Board 4 TEST UNIT

Board 2 signal	Conn.#	Board 4 signal	Conn. #	
SRG1Cs	B2-JP31(4)	SRG1	B4-J1(5)	CON3(36)
SRG2Cs	B2-JP31(3)	SRG2	B4-J1(6)	CON3(35)
SRG3Cs	B2-JP31(2)	SRG3	B4-J1(7)	CON3(34)
SRGsC	B2-JP31(1)	SRGs	B4-J1(14)	CON3(27)
ABGCc	B2-JP42	ABGC	B4-J1(8)	CON3(33)
M_/WAIT	B2-J40(2)	WAIT	B4-J1(15)	CON3(26)
PCSG_STOP_G2	B2-JP20	STOP	B4-J1(16)	CON3(25)
PCSG_STOP	B2-J16(16)	PCSG_STS_spare	B4-JP3	CON3(24)
AD_CS	B2-JP33(10)	AD_CS	B4-J1(4)	CON3(37)
NAND_Spare_A	B2-JP97	PCSG_CTR_spare	B4-JP1	CON3(23)

Total: 10 wires B2 to B4

****

Board 2 - Boards 5

Board 2 signal	Conn.#	Boards 5 signal	Conn. #	TEST UNIT
NAND_Spare_B	B2-JP96	4/3MHz	B5-JP24(4)	WIRE 10cm
CLOCK_4MHz	B2-JP30	CLOCK_4MHz	B5-JP24(2)	WIRE 10cm
/ADCS	B2-JP41	/ADCS	B5-JP24(1)	CON3(34)
ADC_LSB	B2-J16(1)	AD0	B5-JPADIN0	CON0(30)
	B2-J16(2)	AD1	B5-JPADIN1	CON0(10)
	B2-J16(3)	AD2	B5-JPADIN2	CON0(29)
	B2-J16(4)	AD3	B5-JPADIN3	CON0(9)
	B2-J16(5)	AD4	B5-JPADIN4	CON0(28)
	B2-J16(6)	AD5	B5-JPADIN5	CON0(8)
	B2-J16(7)	AD6	B5-JPADIN6	CON0(27)
	B2-J16(8)	AD7	B5-JPADIN7	CON0(7)
	B2-J16(9)	AD8	B5-JPADIN8	CON0(38)
	B2-J16(10)	AD9	B5-JPADIN9	CON0(18)
	B2-J16(11)	AD10	B5-JPADIN10	CON0(37)
ADC_MSB	B2-J16(12)	AD11	B5-JPADIN11	CON0(17)
AD_/BUSY	B2-J16(13)	AD12	B5-JPADIN12	CON0(36)
M_WAIT	B2-J16(15)	AD13	B5-JPADIN13	WIRE 10 cm
M_/WAIT	B2-J16(14)			CON0(16)

Total: 17 wires B2 to B5

****

Board 2 - Board 6

0 wires

****

Board 2 - Boards 7

Board 2 signal	Conn.#	Boards 7 signal	Conn. #	TEST UNIT
ABGC_EN	B2-JP31(5)	HK2 (ABGC_EN)	B7-JP32(7)	CON1(35)
ABGI_in	B2-JP35	HK2 (ABGI)	B7-JP32(6)	CON1(16)
MUX_EN0	B2-JP51	HK2 (MUX_EN0)	B7-JP32(4)	CON1(17)
MUX_EN1	B2-JP33(1)	HK1 (MUX_EN1)	B7-JP31(4)	CON1(9)
MUX_EN2	B2-JP45(2)	LED (MUX_EN2)	B7-JP33(5)	CON1(32)
MUX_EN3	B2-JP45(1)	LED (MUX_EN3)	B7-JP33(6)	CON1(12)
DAC1_L	B2-JP33(9)	HK1 (DAC1_L)	B7-JP31(7)	CON1(27)
S_CLK	B2-JP33(8)	HK1 (S_CLK)	B7-JP31(5)	CON1(28)
S_IN	B2-JP33(7)	HK1 (S_IN)	B7-JP31(6)	CON1(8)
TEMP_EN	B2-JP33(6)	HK2 (TEMP_EN)	B7-JP32(8)	CON1(15)
ANAL_EN	B2-JP33(5)	HK1 (ANAL_EN)	B7-JP31(8)	CON1(7)
DAC_STR_A	B2-JP33(4)	HK1 (DAC_STR_A)	B7-JP31(1)	CON1(30)
DAC_STR_B	B2-JP33(3)	HK1 (DAC_STR_B)	B7-JP31(2)	CON1(10)
DAC_STR_C	B2-JP33(2)	HK1 (DAC_STR_C)	B7-JP31(3)	CON1(29)

Total: 14 wires B2 to B7

****

Board 2 - Host Connector NOT in Test Unit

Board 2 signal	Conn.#	Host Conn. signal	Pin #
MAG_OUT	B2-JP28(1)	MAG_OUT	HC(12)
X_ER_OUT	B2-JP28(2)	X_ER_OUT	HC(10)
Y_ER_OUT	B2-JP28(3)	Y_ER_OUT	HC(9)
DGND	B2-JP101	SignalGND	HC(14)

Total: 4 wires B2 to Host Conn.

****

Board 2 - N.C.

Board 2 signal	Conn.#
ADC_TEST	B2-JP89
AMP1_OUT	B2-JP18(2)
INT_BUFF_OFS_t	B2-JP90
UmIAG_t	B2-JP91
UmABG_t	B2-JP92
U-ABG_t	B2-JP93
POWER_THR	B2-JP88(1..11)
/INT_SAMPLE	B2-JP86
M_/WAIT	B2-J16(14)
PCSG_STOP_G2	B2-J16(17)
DGND	B2-J16(18)
M_WAIT	B2-JP40(1)
DGND	B2-JP24(5)
DGND	B2-JP76(3)
DGND	B2-JP31(6)
DGND	B2-JP45(3)

Total: 26 wires are N.C.

```
*****************************************************
                                                Board 2

Total:  79 wires from B2
                   0 to ED
                  14 to B1
                   (6 B2 internal)
                  20 to B3
                  10 to B4
                  17 to B5
                   0 to B6
                  14 to B7
                   4  to Host Conn.
                ********************
                  26 are N.C.
                  6  are jumpers (B2)
```

BOARD 2 ERRATUM

Differences in between schematic (interconnect list) and silkscreen

NO JP63 and JP88 on board

replase	with	pin
JP63(1)	V-	(1)
JP63(2)	VCCA	
JP63(3)	V+	(1)
JP63(4)	DGND	(1)
JP63(5)	VCC	(1)
JP63(6)	V-c	
JP63(7)	AGND	(1)
JP63(8)	V+c	(1)

There should be no connections to JP88(1)..(11)

Place jumper on GNDJMP pin 1 and 2

****
BOARD 3

****

Board 3 - External Devices (LEDs) (see fig. 4)

Board 3 signal	Conn.#	LED
	B3-D18(1)	+
	B3-D18(2)	GND
	B3-D19(1)	+
	B3-D19(2)	GND
	B3-D20(1)	+
	B3-D20(2)	GND
	B3-D21(1)	+
	B3-D21(2)	GND

TEST UNIT

use wires 40cm one color for GND
separate colors for each + signal
24 gauge

Total: 8 wires form B3 to LEDs

****

Board 3 - Board 1

18 wires listed for Board 3 interconnection

****

Board3 - Board 2

20 wires listed for Board 2 interconnection

****

Board 3 internal connections - Board modifications

U41 pin 4 connect on solder side to U41 pin 2

U45 pin 1 connect on solder side to U45 pin 3 (see Drawing).
Long leads of U45 can be used to make connections.

```
                         o pin 3
                         |
pin 4    o       |       o pin 2
pin 5    o       _____X pin 1

pin 6            o

                 Bottom View
```

Total: 2 wires (4 pins) connected on B3

****

Board 3 - Board 4 NOT in Test Unit

Board 3 signal	Conn.#	Board 4 signal	Conn. #
U+5DM	B3-JP54(2)	VCC	B4-JP7(4)
DGNGM	B3-JP54(1)	GND	B4-JP7(1)

Total: 2 wires form B3 to B4

****

Board 3 - Board 5

Board 3 signal	Conn.#	Board 5 signal	Conn. #	Test Unit
U+5DF	B3-JP53(2)	+5V	B5-J8(1)	
DGNDF	B3-JP53(1)	GND?	B5-J8(4)	
To_ADCLatch_AD14	B3-JP56	AD14	B5-JPADIN14	CONO(35)
To_ADCLatch_AD15	B3-JP82	AD15	B5-JPADIN15	CONO(15)

Total: 4 wires to Board 5

****
Board 3 - Board 6

0 wires

****
Board 3 = Board 7

Board 3 signal	Conn.#	Board 7 signal	Conn. #	TEST UNIT
PC_ON	B3-JP48	LED (EXT_OUT2)	B7-JP33(8)	CON1(31)
LED1	B3-JP23(4)	LED (LED1)	B7-JP33(1)	CON1(34)
LED2	B3-JP23(3)	LED (LED2)	B7-JP33(2)	CON1(14)
LED3	B3-JP23(2)	LED (LED3)	B7-JP33(3)	CON1(33)
LED4	B3-JP23(1)	LED (LED4)	B7-JP33(4)	CON1(13)
To_HS8255	B3-JP85	LED (EXT_OUT1)	B7-JP33(7)	CON1(11)
TO_26C32	B3-JP59(1)	Rx+	B7-JP30(1)	N.C.
TO_26C32	B3-JP59(2)	Rx-	B7-JP30(3)	N.C.
TO_26C31	B3-JP60(1)	Tx+	B7-JP29(1)	N.C.
TO_26C31	B3-JP60(2)	Tx-	B7-JP29(2)	N.C.

Total: 10 wires to Board 7

****

Board 3 - Host Connector NOT in Test Unit

Board 2 signal	Conn.#	Host Conn. signal	Pin #
PWR_RTR	B3-JP49(1)	PWR+28	HC(2)
PWR+28	B3-JP49(3)	PWR_RTR	HC(1)
Dinp	RED_S B3-JP55	Red4DegS	HC(4)
Dinp	AQ_EN B3-JP83	AQ_EN	HC(8)
Dout	B3-JP84	STAR_PRESENT	HC(11)
RS_422_INPS	B3-JP58(2)	RS_422_INPs+	HC(7)
RS_422_INPS	B3-JP58(1)	RS_422_INPs-	HC(13)
RS_422_OUTS	B3-JP61(2)	RS_422_OUTs+	HC(5)
RS_422_OUTS	B3-JP61(1)	RS_422_OUTs-	HC(6)

Total: 9 wires from B3 to Host Conn.

****

Board 3 - N.C.

Board 2 signal	Conn.#
/INH_PWR	B3-JP49(2)

Total: 1 wire on B3 is N.C.

```
************************************************************

Total:  71 wires ; 1 N.C.
                8  to ED - LEDs
                18 to B1
                20 to B2
                ( 2 internal wires on B3 )
                2 to B4
                4 to B5
                0 to B6
                10 to B7
                9  to Host Connector
        ****************************
                1  is N.C.

************************************************************
```

****

BOARD 4

****

Board 4 - ED

0 wires

****

Board 4 - Board 1

8 wires listed for Board 1 interconnections

****

Board 4 - Board 2

10 wires listed for Board 2 interconnections

****

Board 4 - Board 3

2 wires listed for Board 3 interconnections

****

Board 4 - Board 5

Board 4 signal	Conn.#	Board 5 signal	Conn. #
D0	B4-JP4(1)	D0	B5-JPD0(3)
.	.	.	.
.	.	.	.
D15	B4-JP4(16)	D15	B5-JPD15(3)
A1	B4-JP5(1)	A1	B5-JPA1(3)
.		.	.
.		.	.
A7	B4-JP5(7)	A7	B5-JPA7(3)
/WGCS	B4-JP5(8)	/WGCS	B5-JP12(1)
IOWC	B4-JP5(9)	/WR	B5-JP12(3)
IORC	B4-JP5(10)	/RD	B5-JP12(4)
RESIN	B4-JP5(11)	RESET	B5-JP12(2)
BCKL	B4-JP6(3)	4MHz	B5-JP12(5)
INTA	B4-JP6(2)	WGINT	B5-JP12(6)
INTB	B4-JP6(1)	WGTIM	B5-JP12(7)

Total: 30 wires form B4 to Board 5

****

Board 4 - Board 6

0 wires

****

Board 4 - Board 7

0 wires

****

Board 4 - HC

0 wires

****

Board 4 - N.C.

GND	B4-J1(17)
GND	B4-J1(18)
PCSG_ADR_spare	
	B4-JP2
PCSG-GND	B4-JP(1)
PCSG_GND	B4-JP(2)
VCC	B4-JP7(3)
GND	B4-JP7(2)

Total: 7 wires are N.C. on B4

****

Board 4

Total: 50 wires; 7 N.C.

```
 0 to ED
 8 to B1
10 to B2
 2 to B3
30 to B5
 0 to B6
 0 to B7
 0 to HC
************
 7 are N.C.
```

****

BOARD 5

****

Board 5 - ED

0 wires

****

Board 5 - Board 1

0 wires

****

Board 5 - Board 2

17 wires listed for Board 2 interconnections

****

Board 5 - Board 3

4 wires listed for Board 3 interconnections

****

Board 5 - Board 4

30 wires listed for board 4 interconnections

****

Board 5 - Board 6

Board 5 signal	Conn.#	Board 6 signal	Conn. #
A0	B5-JPA0(1)	A0	B6-JP2(1)
.	.	.	.
.	.	.	.
A15	B5-JPA15(1)	A15	B6-JP2(16)
D0	B5-JPD0(1)	D0	B6-JP3(1)
.	.	.	.
.	.	.	.
D15	B5-JPD15(1)	D0	B6-JP3(16)
/WR	B5-JP7(2)	/WR	B6-JP1(2)
/RD	B5-JP7(3)	/RD	B6-JP1(3)
/RAML0	B5-JP7(4)	/RAML0	B6-JP1(4)
/RAMH0	B5-JP7(5)	/RAMH0	B6-JP1(5)
/RAMH1	B5-JP7(6)	/RAMH1	B6-JP1(6)
/RAML1	B5-JP7(7)	/RAML1	B6-JP1(7)
/WREE	B5-JP7(8)	/WREE	B6-JP1(8)
/ROM0	B5-JP7(9)	/ROM0	B6-JP1(9)
/ROM1	B5-JP7(10)	/ROM1	B6-JP1(10)
VCC	B5-JP17(1)	VCC	B6-JP4(1)
GND	B5-JP17(4)	GND	B6-JP4(4)

Total: 43 wire B5 to B6

****

Board 5 - Board 7

Board 5 signal	Conn.#	Board 7 signal	Conn. #
A0	B5-JPA0(2)	A0	B7-JP26(1)
.	.	.	.
.	.	.	.
A15	B5-JPA15(2)	A15	B7-JP26(16)
D0	B5-JPD0(2)	D0	B7-JP27(1)
.	.	.	.
.	.	.	.
D15	B5-JPD15(2)	D0	B7-JP27(16)
/WR	B5-JP41	/WR	B7-JP28(1)
/RD	B5-JP42	/RD	B7-JP28(2)
/UARTCS	B5-JP43	/UARTCS	B7-JP28(3)
/PPICS	B5-JP44	/PPICS	B7-JP28(4)
BAUDCLK	B5-JP45	BAUDCLK	B7-JP28(5)
UARTIN	B5-JP46	UARTIN	B7-JP28(6)
UARTOUT	B5-JP47	ARTOUT	B7-JP28(7)
4MHz	B5-JP48	4MHz	B7-JP28(8)
RESET	B5-JP49	RESET	B7-JP28(9)
/RESET	B5-JP410	/RESET	B7-JP28(10)
DT/R	B5-JP411	DT/R	B7-JP28(11)
/WATCHDOG	B5-JP412	/WATCHDOG	B7-JP28(12)
/DEN	B5-JP413	/DEN	B7-JP28(13)
VCC	B5-JP18(1)	VCC	B7-JP25(1)
GND	B5-Jp18(4)	GND	B7-JP25(4)

Total: 47 wires B5 to B7

Board 5 - N.C.

Board 5 signal	Conn.#
A0	JPA0(3)
A8	JPA8(3)
A9	JPA9(3)
N.C.	JP12(8)
VCC	JP8(2)
GND	JP8(3)
VCC	JP17(2)
GND	JP17(3)
A16	JP7(11)
/WR	JP7(1)
4/3MHz	JP414
VCC	JP18(2)
GND	JP18(3)
N.C.	Jp18(14)

Total: 14 wires are N.C.

****

Board 5

Total: 141 wires; 2 internal

0 to ED
0 to B1
17 to B2
4 to B3
30 to B4
(2 internal B5)
43 to B6
47 to B7
0 to HC
****
14 are N.C.

Board 6

Board 6 - ED

0 wires
****

Board 6 - Board 1

0 wires
****

Board 6 - Board 2

0 wires
****

Board 6 - Board 3

0 wires
****

Board 6 - Board 4

0 wires
****

Board 6 - Board 5

43 wires listed for board 5 interconnections
****

Board 6 - Board 7

0 wires
****

Board 6 - HC

0 wires
****

Board 6 - N.C.

Board 6 signal	Conn.#
VCC	JP4(2)
GND	JP4(3)
N.C.	JP1(11)

Total: 3 wires are N.C.
****

Board 6

Total: 43 wires

0 to ED
0 to B1
0 to B2
0 to B3
0 to B4
43 to B5
0 to B7

****

3 are N.C.

****

Board 7 - ED

0 wires

****

Board 7 - Board 1

4 wires listed for Board 1 interconnections

****

Board 7 - Board 2

14 wires listed for Board 2 interconnections

****

Board 7 - Board 3

10 wires listed for Board 3 interconnections

****

Board 7 - Board 4

0 wires

****

Board 7 - Board 5

47 wires listed for Board 5 interconnections

****

Board 7 - Board 6

0 wires

****

Board 7 - internal jumpers

Board 7 signal	Conn.#	Board 7 signal	Conn. #
VCC	B7-JP30(10)	/DSR+	B7-JP30(9)
GND	B7-JP30(12)	/DSR-	B7-JP30(12)

Total: 4 signal are internally connected on B7

****

Board 7 - HC

0 wires

****

Board 7 - N.C.

Board 7 signal	Conn.#
A0	JP37(1)
.	.
.	.
A15	JP37(16)
D0	JP38(1)
.	.
.	.
D15	JP38(16)
VCC	JP25(2)
GND	JP25(3)
VCC	JP30(2)
GND	JP30(4)
VCC	JP30(6)
GND	JP30(8)
RTS+	JP29(3)
RTS-	JP29(4)
/DTR+	JP29(5)
/DTR-	JP29(6)
CTS+	JP30(5)
CTS-	JP30(7)

Total: 44 wires are N.C.

****

Board 7

Total: 75 wires; 2 internal

0 to ED
4 to B1
14 to B2
10 to B3
0 to B4
47 to B5
0 to B6
(4 internal B7)
0 to HC
****
44 are N.C.

Host Connector

HC - ED

HC signal	Conn.#	External Device
Chassis	15	Chassis - Connector screw

Total: 1 wires
****

HC - Board 1

0 wires
****

HC- Board 2

4 wires listed for Board 2 interconnections
****

HC - Board 3

9 wires listed for Board 3 interconnections
****

HC - Board 4

0 wires
****

HC - Board 5

0 wires
****

HC - Board 6

0 wires
****

HC - Board 7

0 wires
****

HC - N.C.

HC signal	Conn.#
Reduced Search 2 deg	3

Tota: 1 pin is N.C.

****

Host Connector

Total: 14 wires; 1 N.C.

```
 1 to ED
 0 to B1
11 to B2
 0 to B3
 0 to B4
 0 to B5
 0 to B6
 0 to B7
****************
 1 is N.C.
```

Test Connectors

Serial Connector

CONS DB9 female

CON Pin#	Signal	Board 7	Signal
1	RX+	B7-JP29(1)	TX+
2	RX-	B7-JP29(2)	TX-
3	TX-	B7-JP30(1)	RX-
4	TX+	B7-JP30(3)	RX-
5	GND	B7-JP30(4)	DGND
6	RTS+	B7-JP30(5)	CTS+
7	RTS-	B7-JP30(7)	CTS-
8	CTS-	B7-JP29(3)	RTS+
9	CTS+	B7-JP29(4)	RTS-

Remarks

Terminal controljumpers on board 7 are in test positions

test position (/DSR- to VCC, /DSR+ to GND)

B7-JP30(9) connect to B7-JP30(8)
B7-JP30(11) connect to B7-JP30(10)

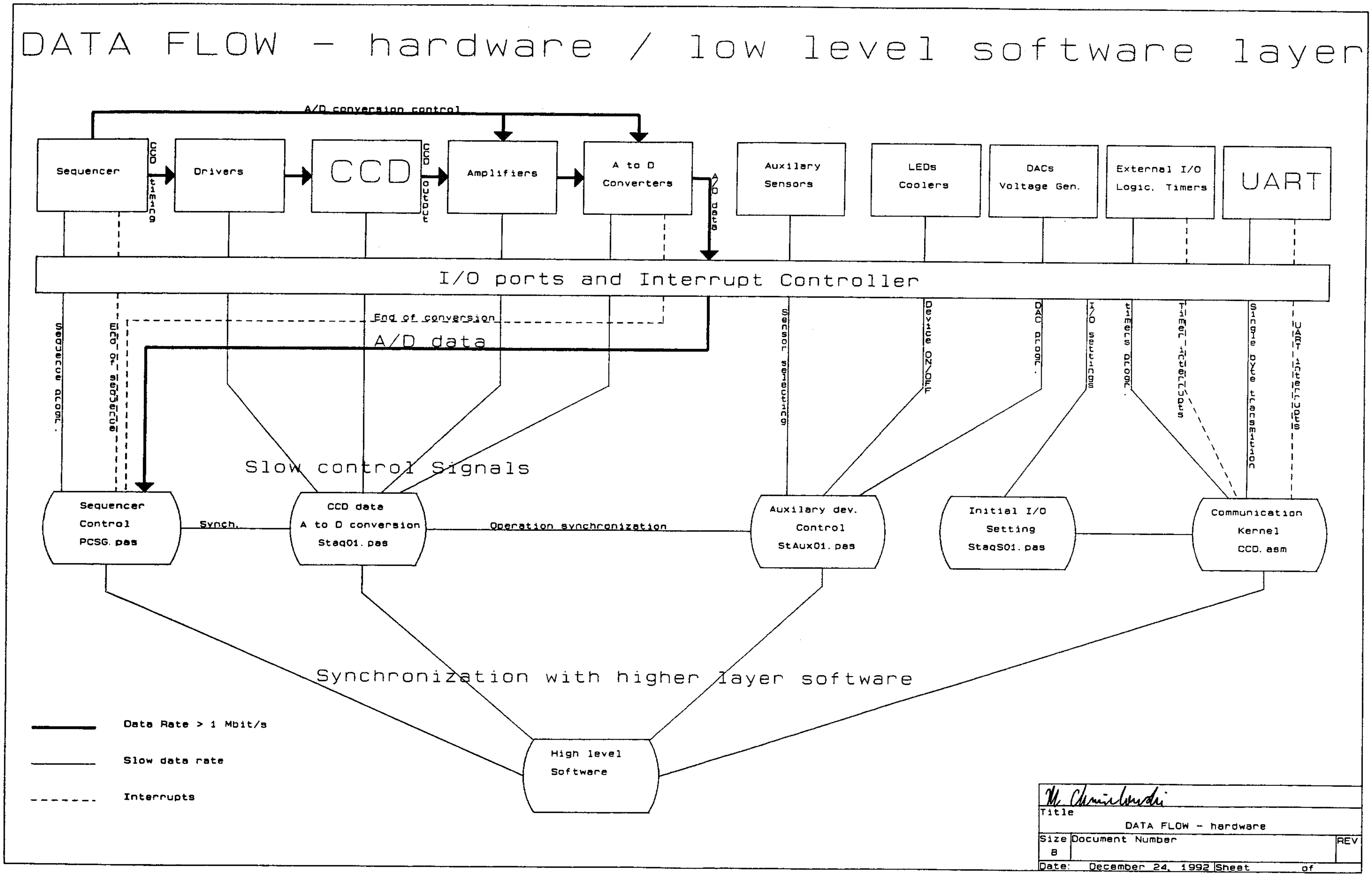
flight position (/DSR+ to VCC, /DSR- to GND)

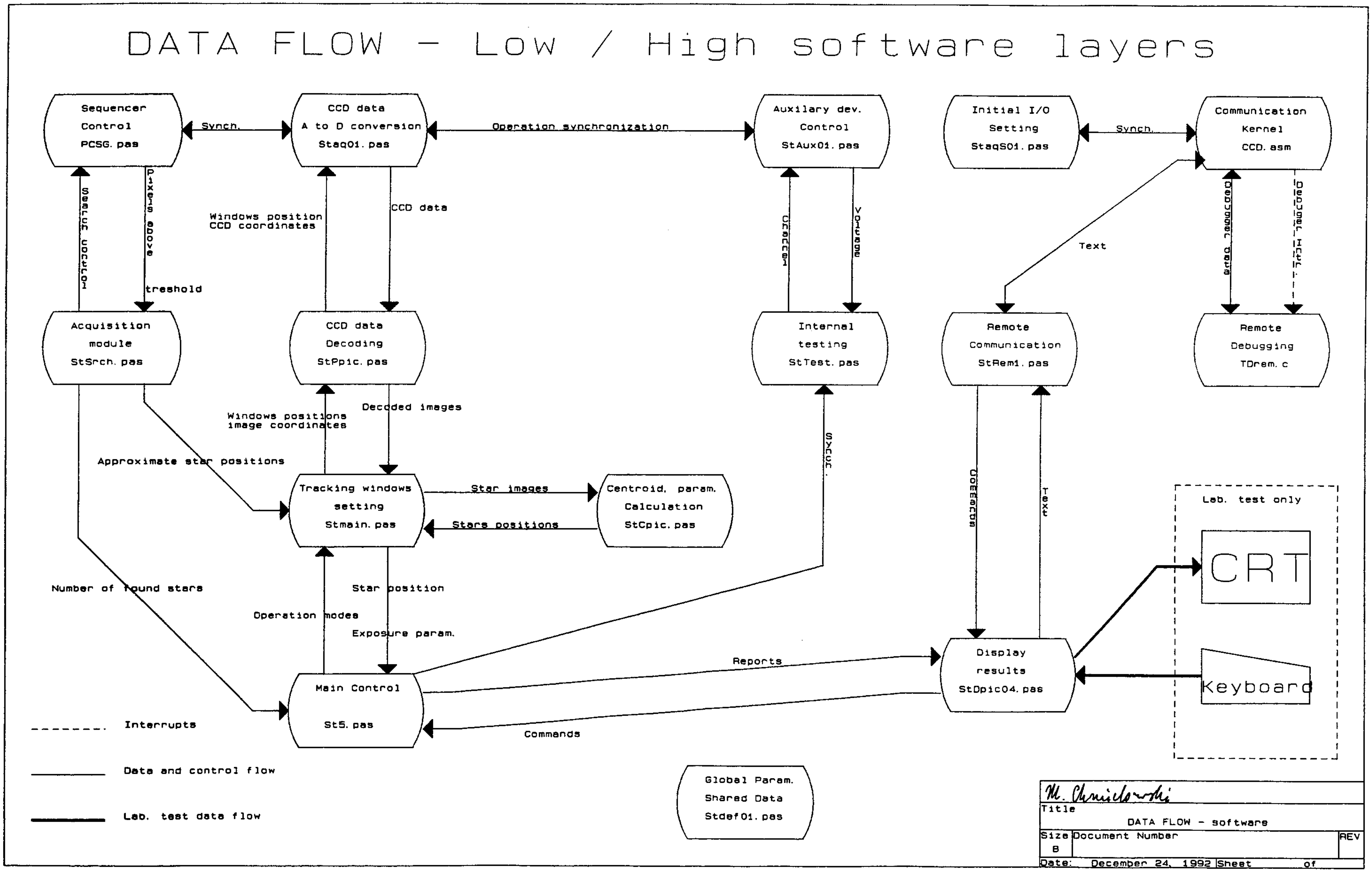
B7-JP30(9) connect to B7-JP30(10)
B7-JP30(11) connect to B7-JP30(8)

Power Connector

CONP DB15 male

CON Pin#	Signal	Board7	Signal
1	+5VDC	B7-JP25(2)	VCC
4	GND	B7-JP25(3)	GND

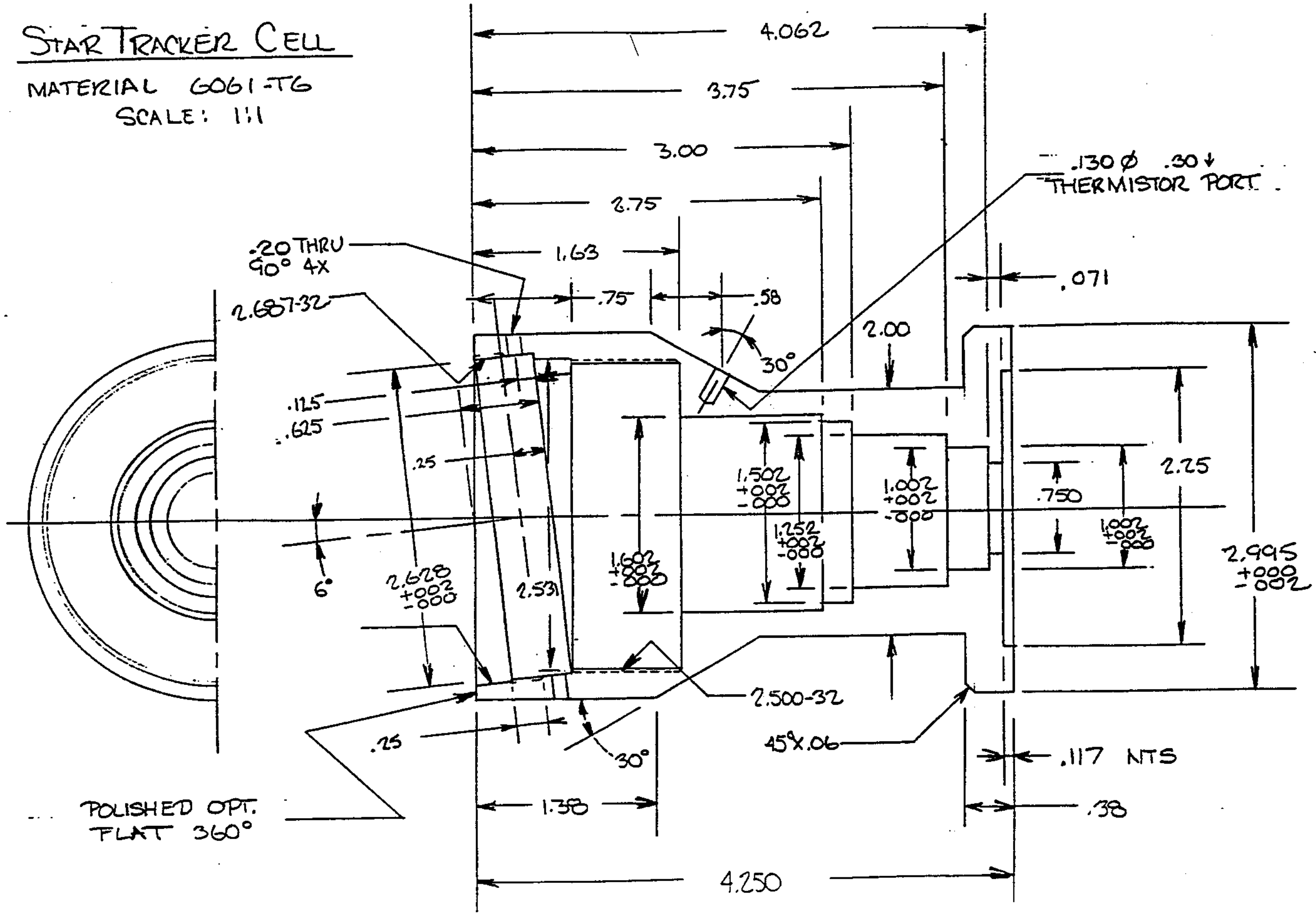

DATA FLOW - hardware / low level software layer
A/D conversion control
Sequencer
CCD timing
Drivers
CCD
CCD output
Amplifiers
A to D Converters
A/D data
Auxilary Sensors
LEDs Coolers
DACs Voltage Gen.
External I/O Logic, Timers
UART
I/O ports and Interrupt Controller
Sequence clock
End of sequence
End of conversion
A/D data
Sensor selecting
Device ON/OFF
DAC clock
I/O settings
Timers clock
Timer interrupts
Single byte transmission
UART interrupts
Slow control Signals
Sequencer Control PCSG.pas
Synch.
CCD data A to D conversion StaQ01.pas
Operation synchronization
Auxilary dev. Control StAux01.pas
Initial I/O Setting StaqS01.pas
Communication Kernel CCD.asm
Synchronization with higher layer software
High level Software
Data Rate > 1 Mbit/s
Slow data rate
Interrupts
Title
DATA FLOW - hardware
Size B
Document Number
REV
Date: December 24, 1992
Sheet of



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

Technical Drawings

B13001.000	LENS CELL
B13001.001	LENS CELL, BOTTOM VIEW
B13002.000	WINDOW RETAINER AND CELL "A" LOCK RING
A13002.100	WINDOW
B13003.000	SPACER, ELEMENTS "E-F", AND CELL, ELEMENT "A"
B13004.000	RETAINER, ELEMENTS "B-F', AND RETAINER, ELEMENT "A"
B13005.000	ELEMENTS "C-D" SPACER AND ELEMENTS "D-E" SPACER
D13006.000	CELL INTERFACE FLANGE
B13007.000	CERAMIC INTERFACE, FRONT, BOTTOM AND SIDE VIEWS
B13008.000	CCD BRIDGE
D13009.000	COVER
C13010.000	BRACKET, STAR TRACKER MOUNTING
A13011.000	CABLE CLAMP, BOTTOM
A13011.001	CABLE CLAMP, TOP
B13012.000	MODIFICATION FLANGE AND CCD - ASSEMBLY - SKETCH

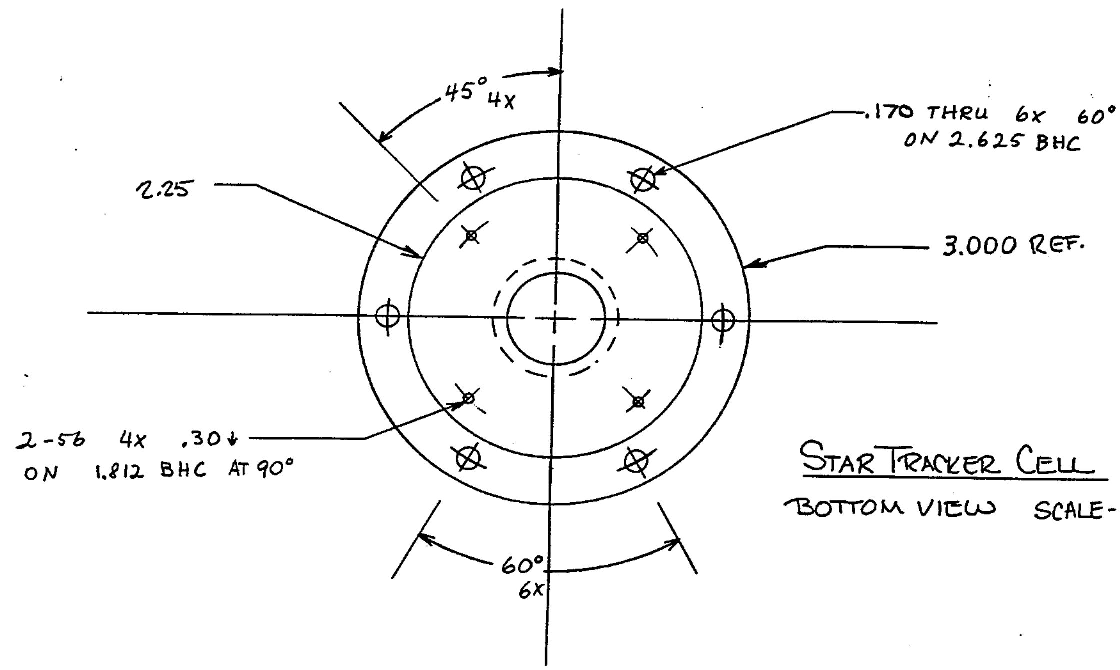


APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	TIM STEEN	FEB. 1992

STAR TRACKER, LENS CELL

DRAWING NO.	SHEET NO.
B 13 001.000	1 OF 2

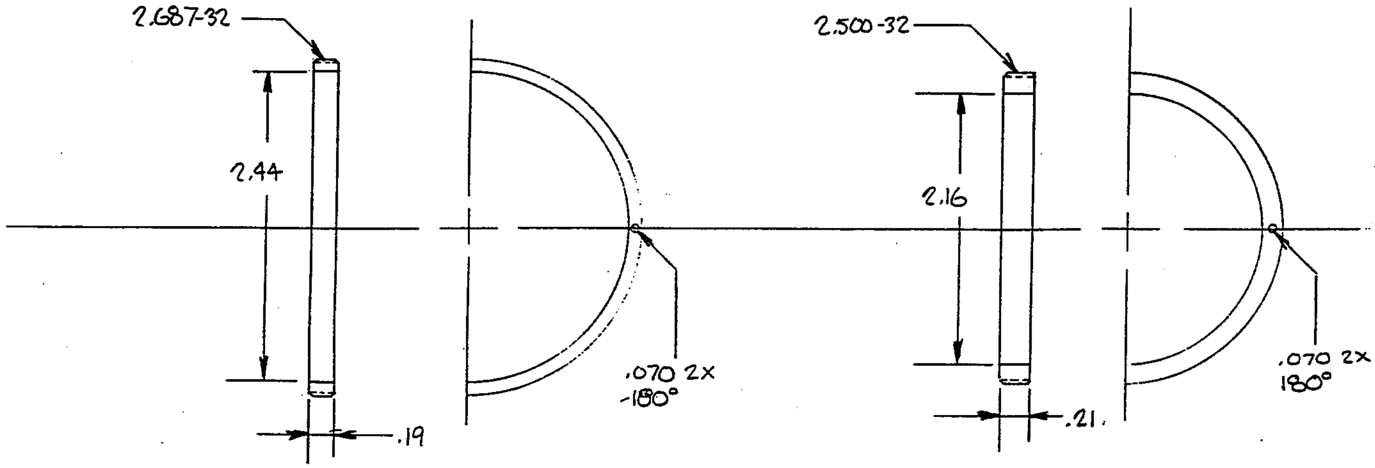


STAR TRACKER CELL

BOTTOM VIEW SCALE- 1:1

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, LENS CELL		
DRAWING NO. B 13 001.001		SHEET NO. 2 OF 2



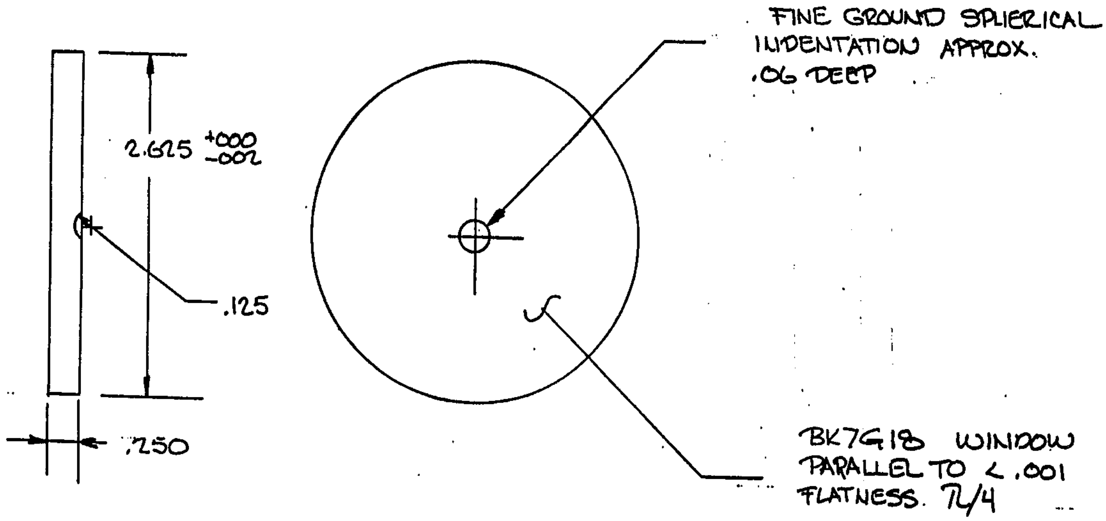
WINDOW RETAINER

MATERIAL 6061-T6 SCALE 1:1

CELL "A" LOCK RING

MATERIAL 6061 T6 SCALE 1:1

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, WINDOW RET. & "A" LOCK RING		
DRAWING NO. B 13 002.000		SHEET NO.



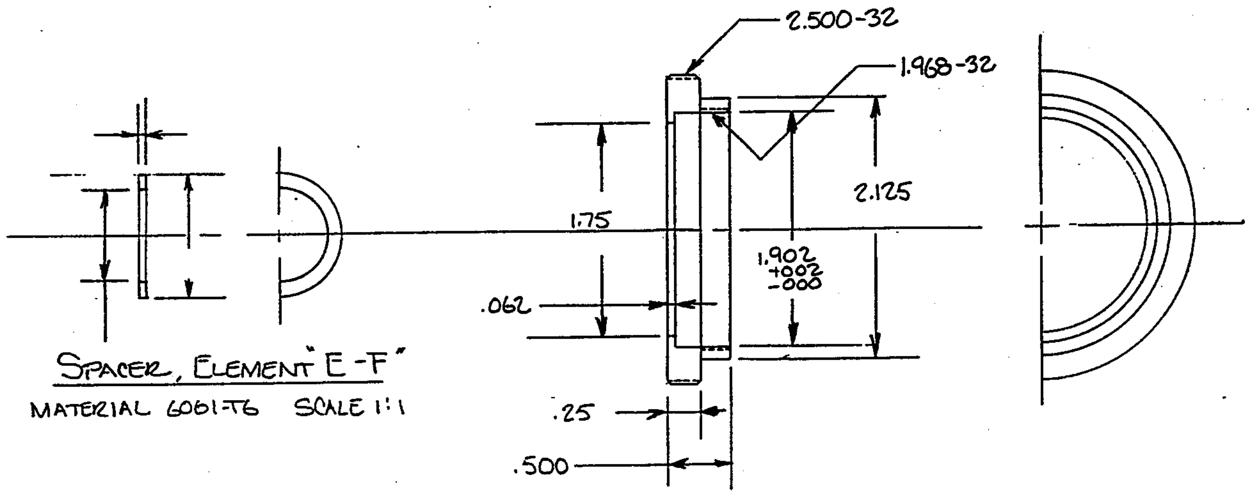
WINDOW, STAR TRACKER

MAT: BK7G18

A 13002.100

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, WINDOW		
DRAWING NO. A 13002.100		SHEET NO.



SPACER, ELEMENT "E-F"
MATERIAL 6061-T6 SCALE 1:1

CELL, ELEMENT "A"
MATERIAL 6061-T6 SCALE 1:1

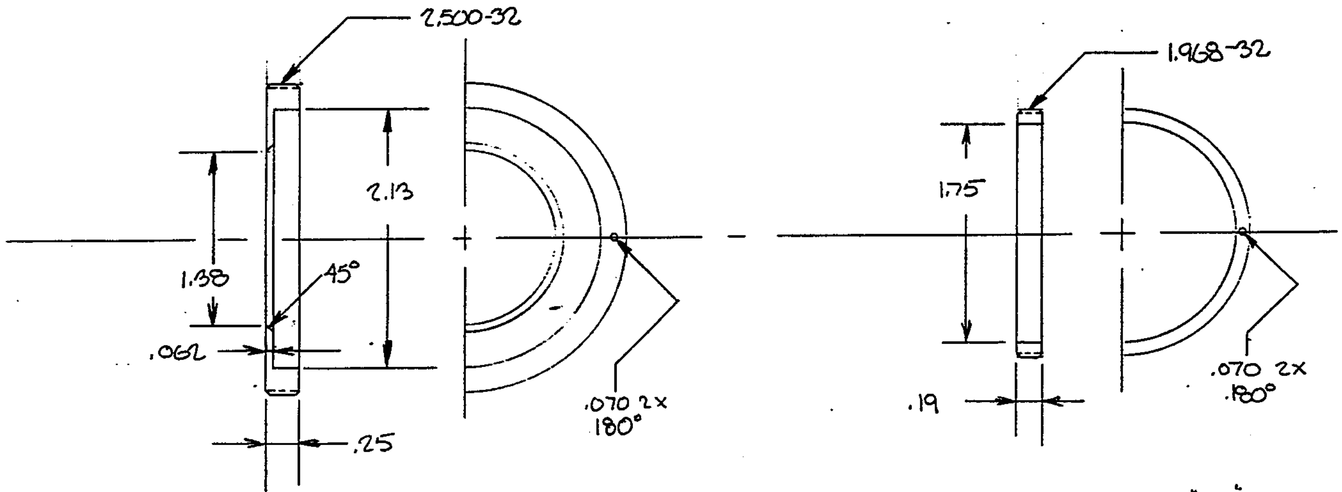
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	JIM STEEN	FEB. 1992

STAR TRACKER, LENS EL. "E-F" & "A"

DRAWING NO.	SHEET NO.
B 13 003.000	



RETAINER, ELEMENTS "B-F"

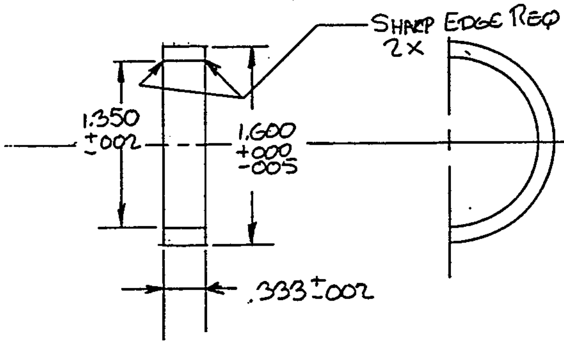
MATERIAL 6061-T6 SCALE 1:1

RETAINER, ELEMENT "A"

MATERIAL 6061-T6 SCALE 1:1

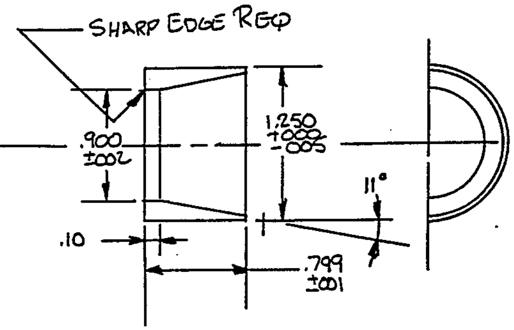
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, RETAINER EL. "B-F" & "A"		
DRAWING NO. 13 004.000		SHEET NO.



ELEMENTS "C-D" SPACER

MATERIAL 6061-T6 SCALE 1:1

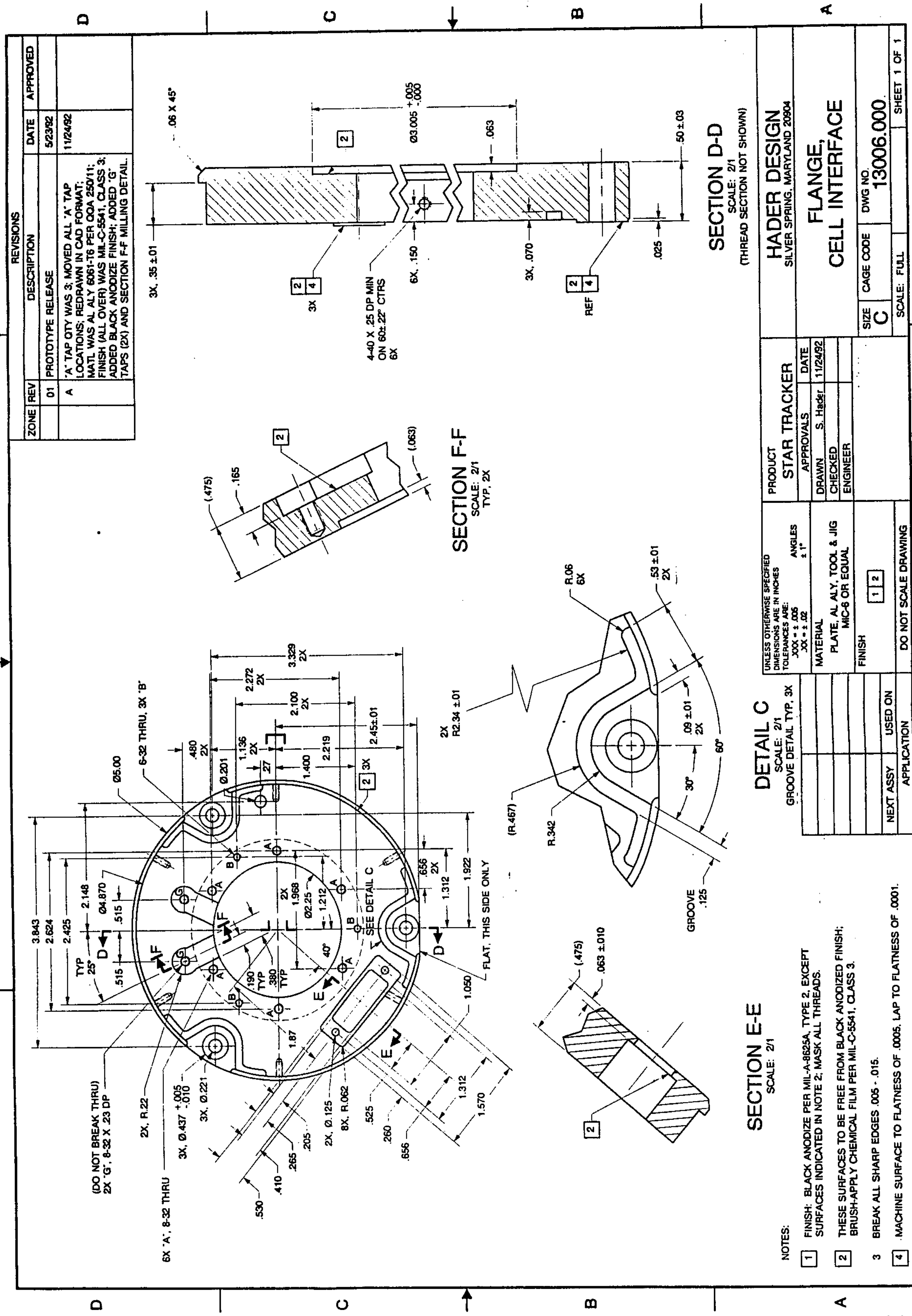


ELEMENTS "D-E" SPACER

MATERIAL INVAR SCALE 1:1

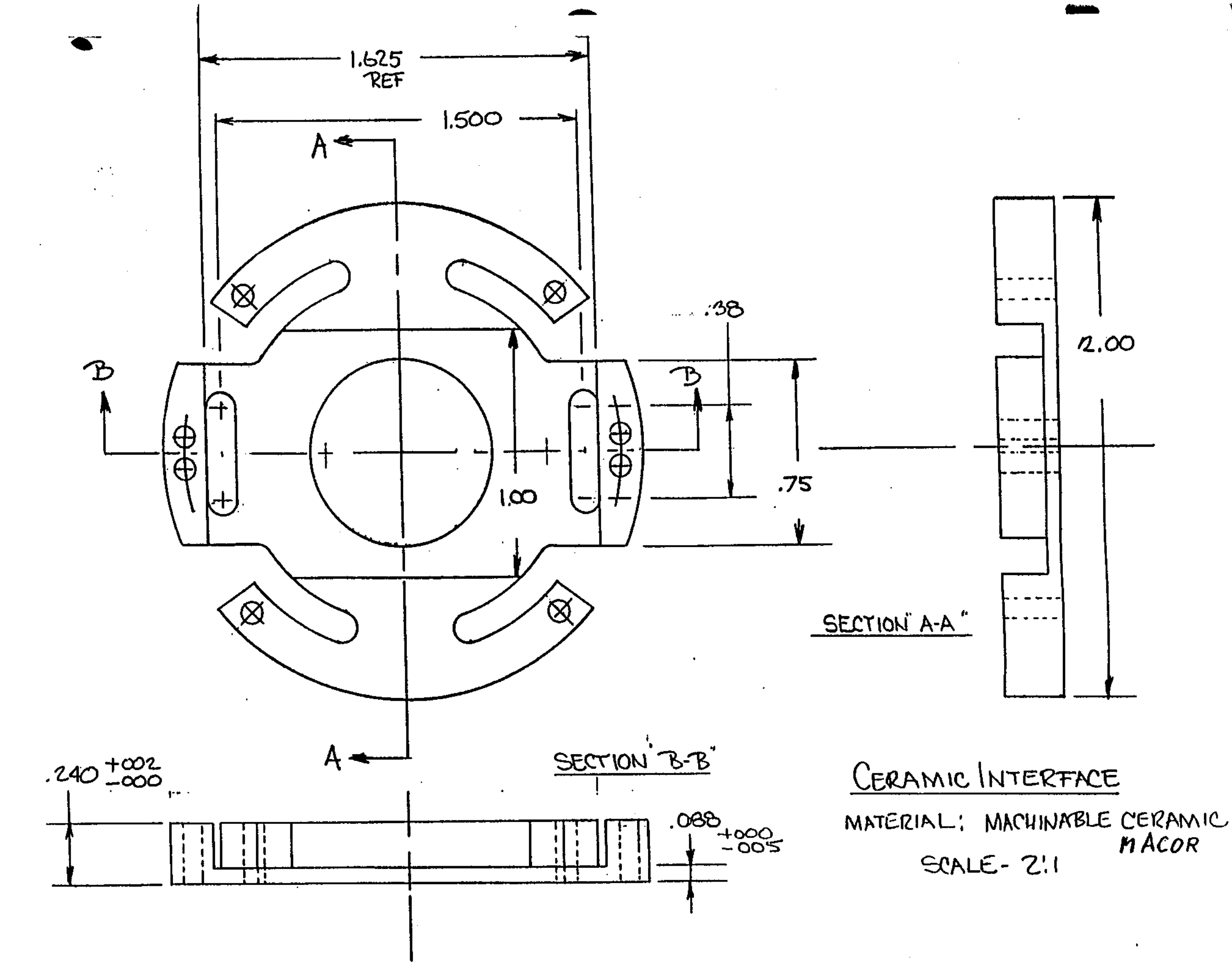
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

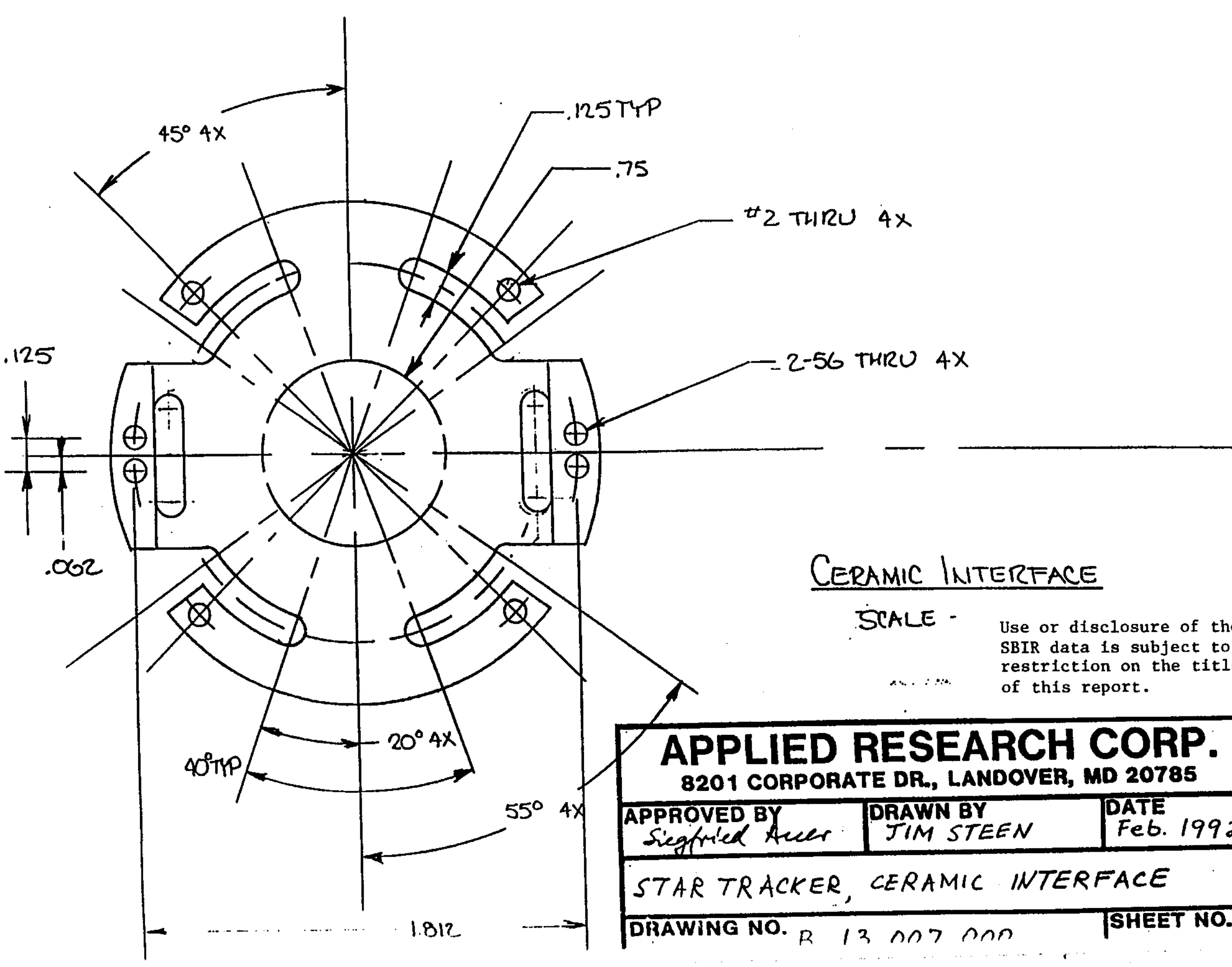
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, SPACER EL. "C-D" & "D-E"		
DRAWING NO. 13 005.000		SHEET NO.



Approved but not signed
by Dr. S. Auer

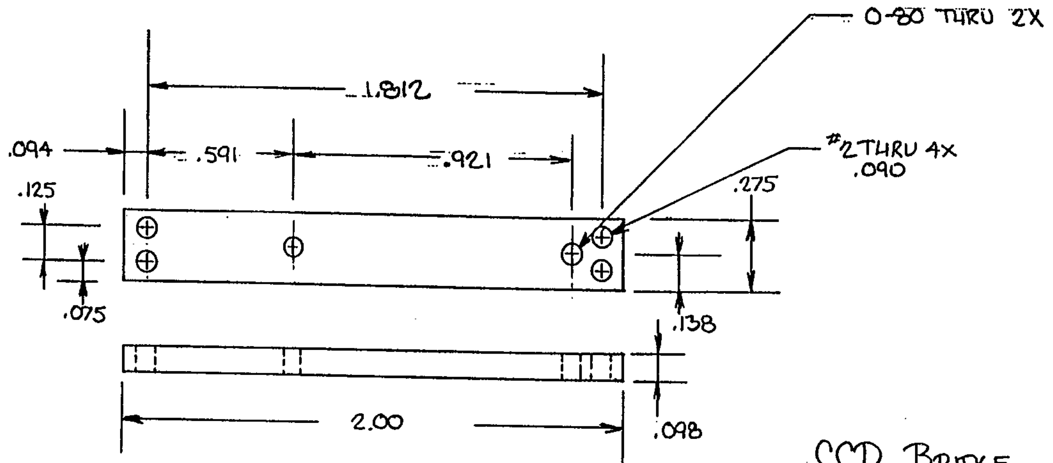
Use or disclosure of these
SBIR data is subject to the
restriction on the title page
of this report.





Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE Feb. 1992
STAR TRACKER, CERAMIC INTERFACE		
DRAWING NO. B 13 007 000		SHEET NO.



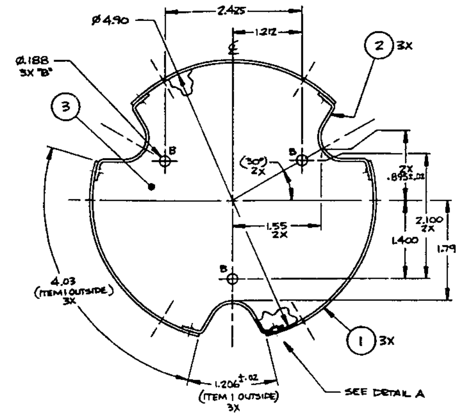
CCD BRIDGE

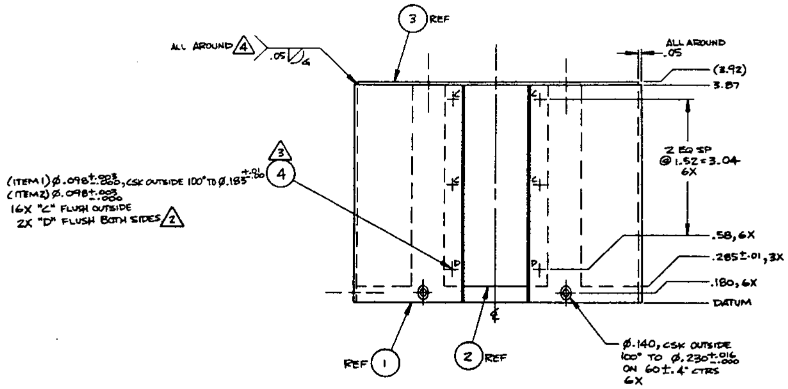
MATERIAL: MOLYBDENUM
SCALE:- 2:1

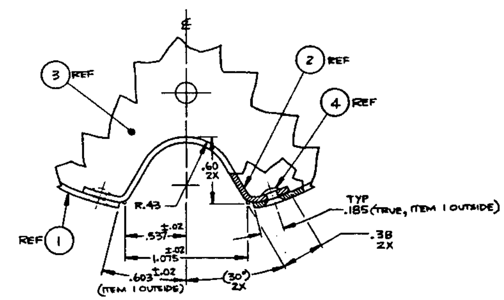
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY JIM STEEN	DATE FEB. 1992
STAR TRACKER, CCD BRIDGE		
DRAWING NO. A 13 008.000		SHEET NO.

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

ZONE	REV	DESCRIPTION	DATE	APPROVED
	-	PROTOTYPE RELEASE	5/23/92	







DETAIL A
SCALE: 2/1
TYP, 3X

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

NOTES:

1. UNLESS OTHERWISE SPECIFIED, ALL BENDS .05 MAX BEND R; REMOVE BURRS AND BREAK SHARP EDGES.
2. 2X "D" ONLY: FABRICATOR OPTION TO USE RIVETS (ITEM 4), SPOTWELD, OR PLUGWELD. RIVETS MUST BE FLUSH (CSK) NEARSIDE & FLUSH OR BELOW FLUSH (CBORE) FARSIDE. GRIND PLUGWELDS FLUSH BOTH SIDES. ATTACHMENT METHOD CHOSEN MUST ENSURE SOUND CONSTRUCTION; NO WARPING, TWISTING, BOWING, OR GAPS ALLOWED.
3. INSTALL RIVETS PER MFGR'S INSTRUCTIONS; DRIVEN HEAD HEIGHT .06 MAX. NO WARPING, TWISTING, BOWING, OR GAPS ALLOWED AT RIVETED JOINTS.
4. CONTINUOUS WELD; NO GAPS OR VOIDS. MIN WELD BUILD-UP ALLOWED INSIDE.

Approved but not signed by Dr. S. Auer

ITEM NO.	PART NO.	QTY	DESCRIPTION	MANUFACTURER/SPEC
4	MS20426B3-4	18	RIVET, SOLID, 100° FLT HD, Ø.094 X .250 LG, AL ALY 5056-H32	ANY
3	-7	1	SHEET, .050 THK, AL ALY 6061-T6	QQA-250/11
2	-6	3	SHEET, .050 THK, AL ALY 6061-T6	QQA-250/11
1	-5	3	SHEET, .050 THK, AL ALY 6061-T6	QQA-250/11

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE: .XXX = ± .005, .XX = ± .02, ANGLES ± 2°

MATERIAL: SEE PARTS LIST

FINISH: CHEMICAL FILM PER MIL-C-5541, CLASS 3

DO NOT SCALE DRAWING

NEXT ASSY. | USED ON — APPLICATION

PRODUCT: STAR TRACKER

APPROVALS		DATE
DRAWN	S. Hader	5/23/92
CHECKED		
ENGINEER		

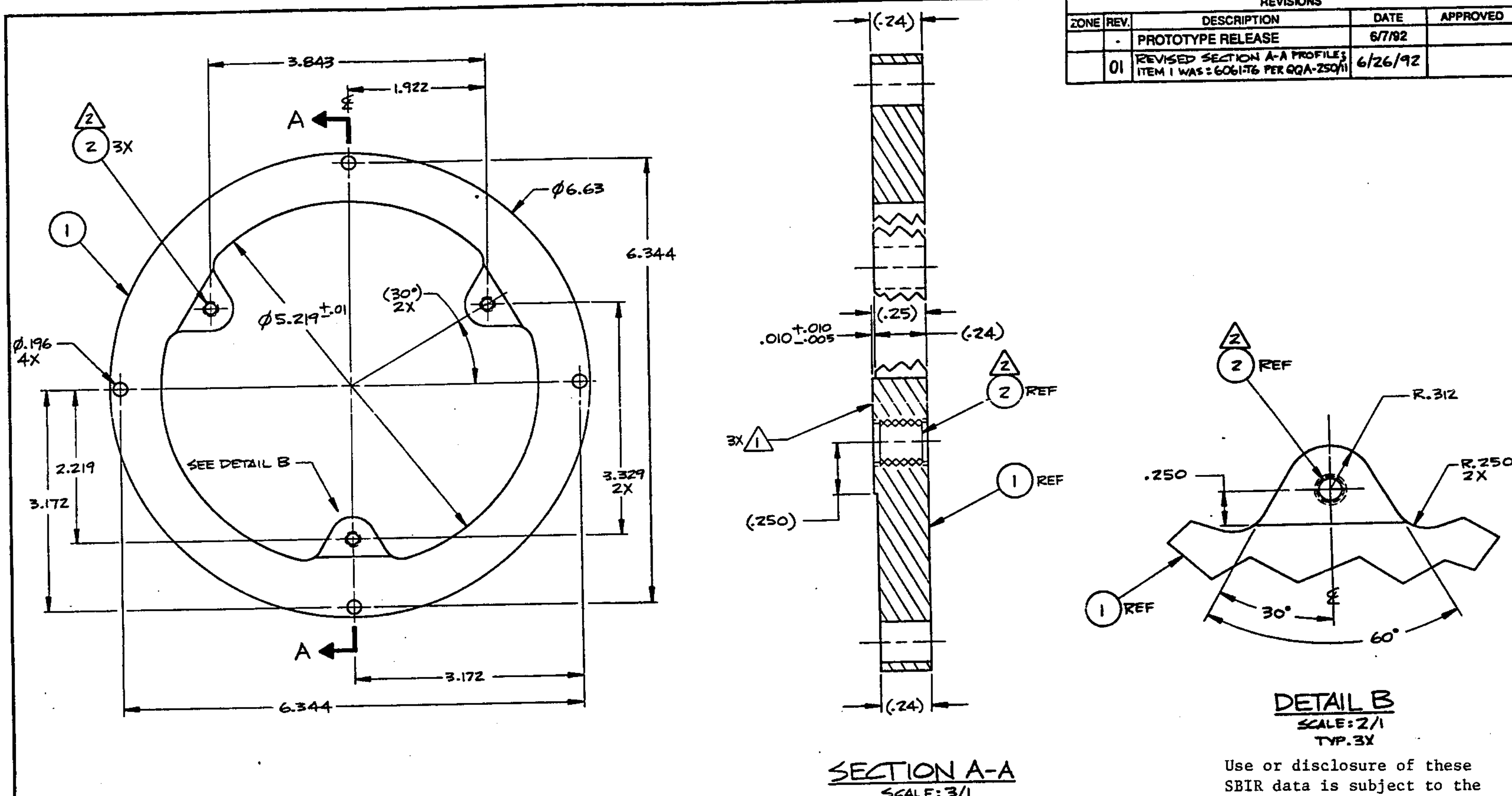
HADER DESIGN
SILVER SPRING, MARYLAND 20904

COVER, STAR TRACKER

SIZE	CAGE CODE	DWG NO.	REV.
D		13009.000	-

SCALE: FULL — SHEET 1 OF 1

REVISIONS				
ZONE	REV.	DESCRIPTION	DATE	APPROVED
	-	PROTOTYPE RELEASE	6/7/92	
	01	REVISED SECTION A-A PROFILES; ITEM 1 WAS: 6061-T6 PER QQA-250/11	6/26/92	



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

Approved but not signed by Dr. S. Auer

NOTES:

1. MACHINE SURFACE TO FLATNESS OF .0005, LAP TO FLATNESS OF .0001.
2. INSTALL THREAD INSERT PER MFGR'S INSTRUCTIONS; NO PART OF INSERT ALLOWED TO PROTRUDE BEYOND SURFACE OF ITEM 1, BOTH SIDES.
3. BREAK ALL SHARP EDGES .005 - .015.

ITEM NO.	PART NO.	QTY	DESCRIPTION	MANUFACTURER/SPEC
2	3591-3CN 0190	3	THREAD INSERT, 10-32 X .190 LG, LOCKING, SST	HELICOIL
1	-5	1	PLATE, .25 THK, AL ALY, TOOL & JIG	MIC-6 OR EQUAL

PARTS LIST

NEXT ASSY.	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE: .XXX = ±.005, .XX = ±.02; ANGLES ±1°

MATERIAL: SEE PARTS LIST

FINISH: CHEMICAL FILM PER MIL-C-5541, CLASS 3

DO NOT SCALE DRAWING

PRODUCT: N/A

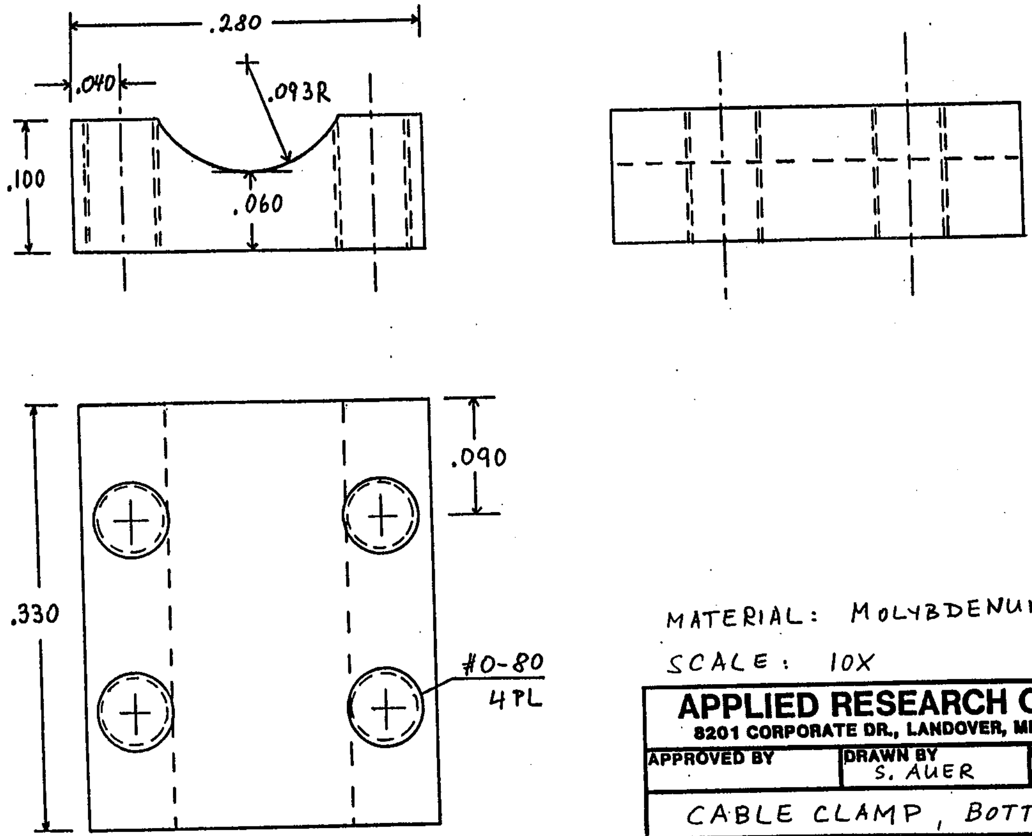
APPROVALS		DATE
DRAWN	S. Hader	6/7/92
CHECKED		
ENGINEER		

HADER DESIGN
SILVER SPRING, MARYLAND 20904

BRACKET, STAR TRACKER MTG

SIZE	CAGE CODE	DWG NO.	REV.
C		13010.000	01

SCALE: FULL — SHEET 1 OF 1

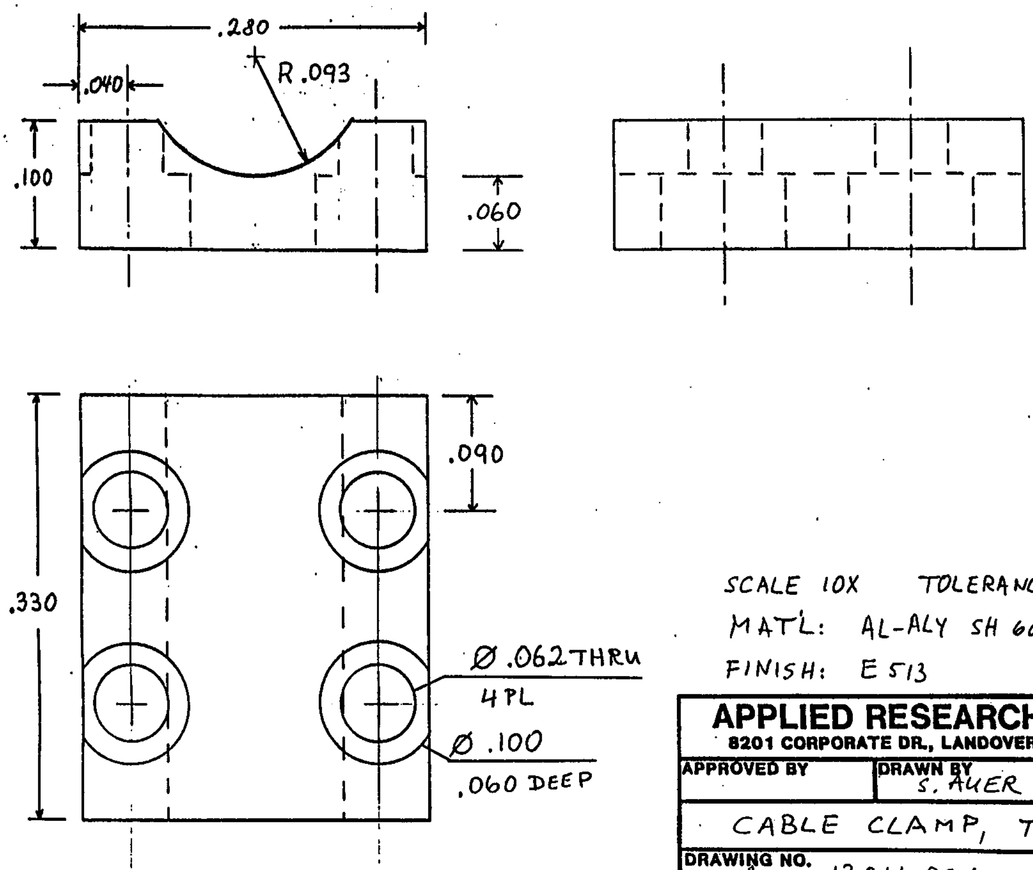


MATERIAL: MOLYBDENUM

SCALE: 10X

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY	DRAWN BY S. AUER	DATE 7-24-92
CABLE CLAMP, BOTTOM		
DRAWING NO. A 13011.000		SHEET NO.

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



SCALE 10X   TOLERANCES .003

MAT'L: AL-ALY SH 6061 -T6

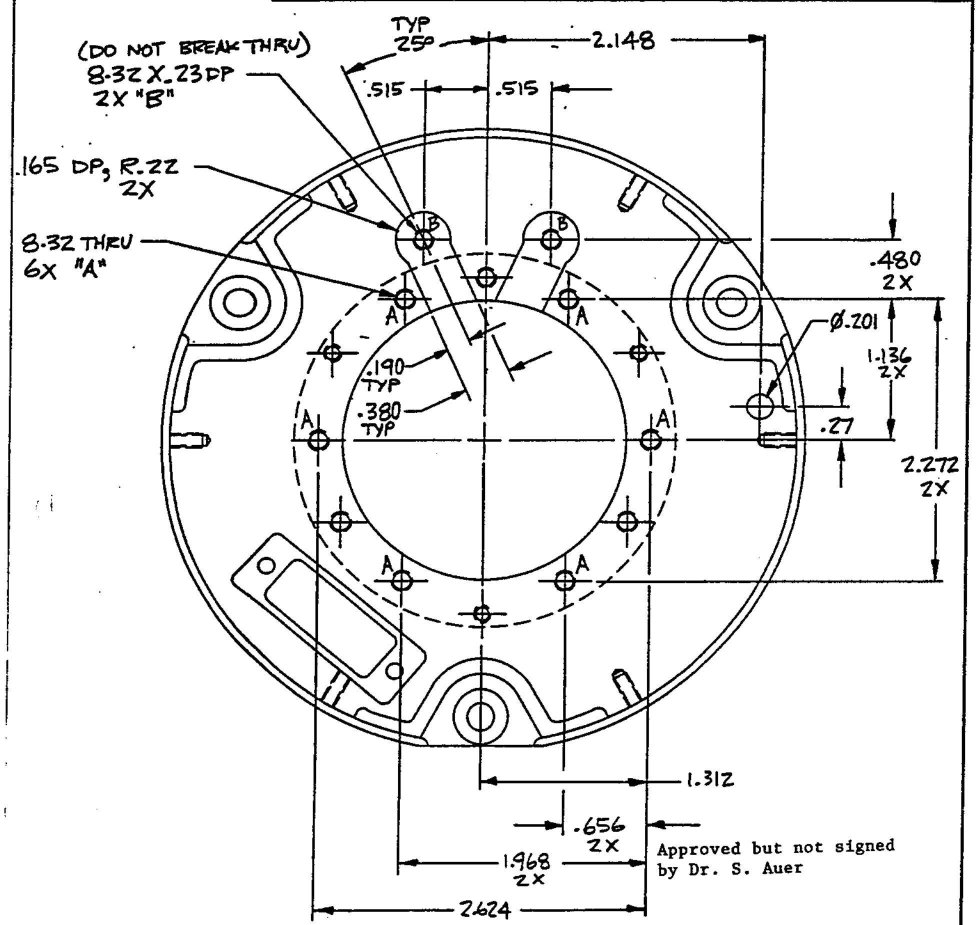
FINISH: E 513

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY	DRAWN BY S. AUER	DATE 8-3-92
CABLE CLAMP, TOP.		
DRAWING NO. A 13011.001		SHEET NO.

FSCM	DWG NO	SHT	REV

REVISIONS

ZONE	REV	DESCRIPTION	DATE	APPROVED



Approved but not signed by Dr. S. Auer

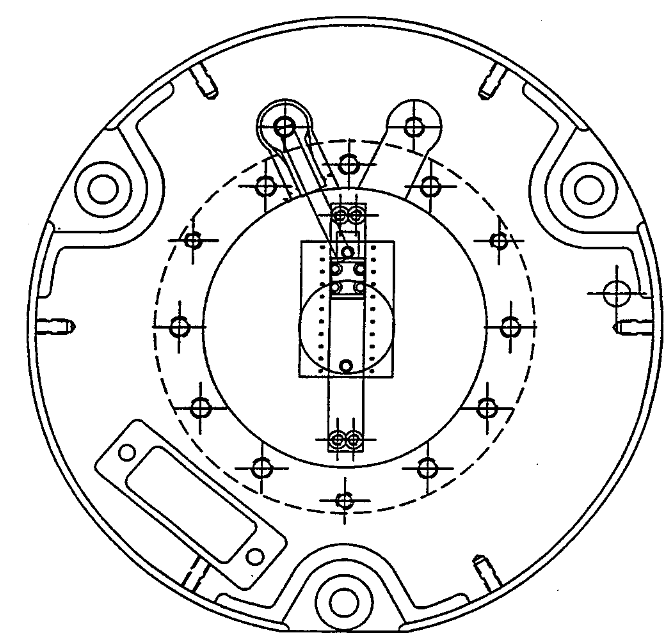
N… E: All dimensions in inches unless otherwise noted.

TOLERANCES	HADER DESIGN SILVER SPRING, MD 20904
.X ±.03; .XX ±.02; .XXX ±.005; ∠ ±1° UNLESS OTHERWISE SPECIFIED	MODIFICATION, CELL I/F FLANGE
DRAWN Steve Hader	A; FSCM NO.; DWG NO. SKETCH; REV -
ISSUED 10/1/92	SCALE 1/1; WEIGHT; SHEET 1 OF 1

FSCM	DWG NO	SHT	REV

REVISIONS

ZONE	REV	DESCRIPTION	DATE	APPROVED



Approved but not signed by Dr. S. Auer

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

NOTE: All dimensions in inches unless otherwise noted.

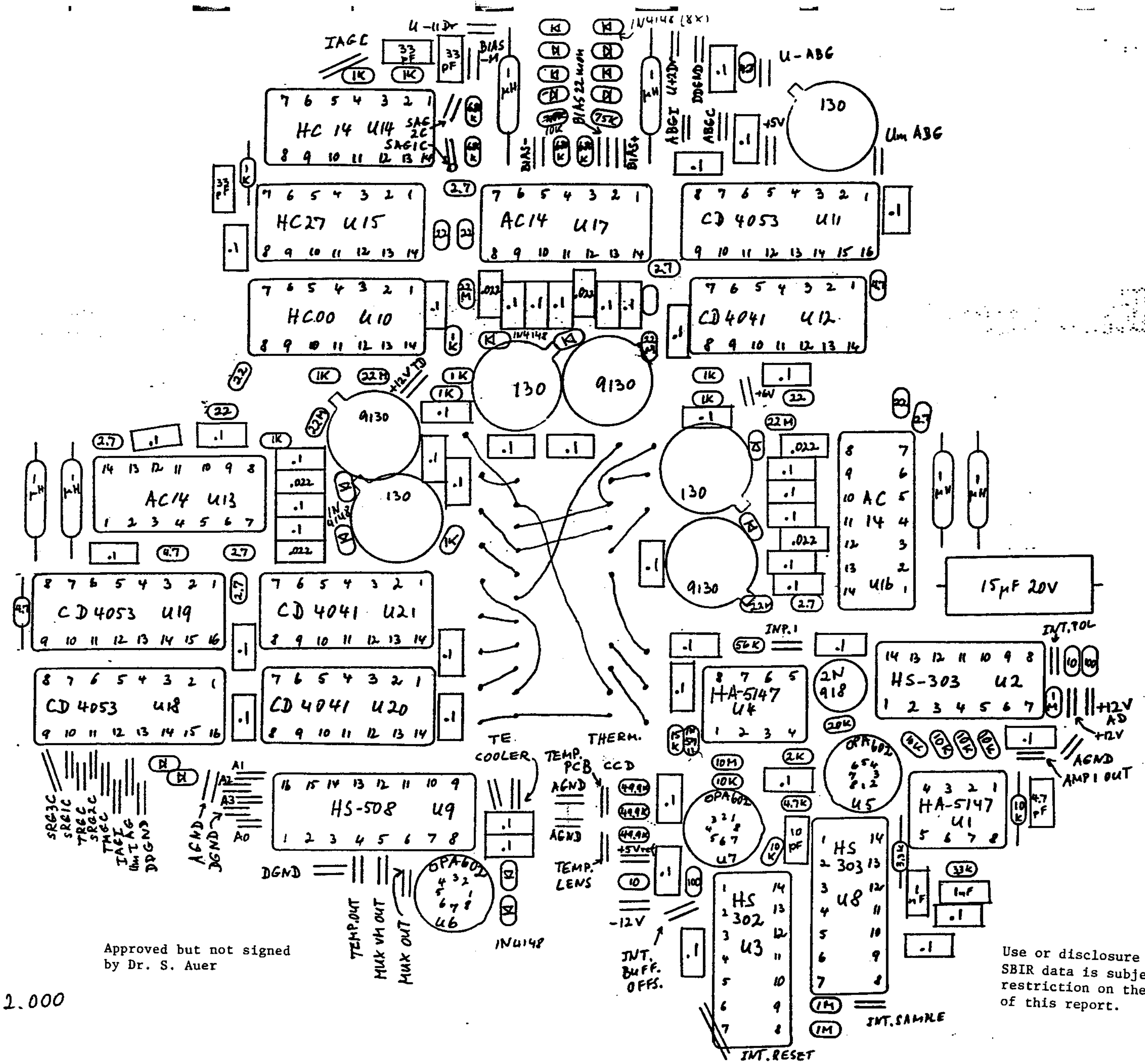
TOLERANCES	HADER DESIGN SILVER SPRING, MD 20904
.X ±.03; .XX ±.02; .XXX ±.005; ∠ ±1° UNLESS OTHERWISE SPECIFIED	MODIFICATION, CELL I/F FLANGE
DRAWN Steve Hader	A; FSCM NO.; DWG NO. SKETCH; REV -
ISSUED 10/1/92	SCALE 1/1; WEIGHT; SHEET 1 OF 1

Electronics Schematics and PCB Layout

B13012.000	PCB 1	ASSEMBLY DWG
B13012.005		CCD ASSEMBLY
B13012.010		ARTWORK LAYER 1 (COMP. SIDE)
B13012.020		2
B13012.030		3
B13012.040		4 (SOLDER SIDE)
B13012.100		FAB DWG
B13012.900		SCHEMATIC, Bds. 1-3
B13012.910		Bd.1a, Dual Slope Int. Amp.
B13012.920		Bd.1b, CCD Drivers
B13013.000	PCB 2	ASSEMBLY DWG
B13013.010		ARTWORK LAYER 1
B13013.020		2
B13013.030		3
B13013.040		4
B13013.100		DRILL DWG
B13013.101		DRILL TABLE
B13013.110		SILKSCREEN
B13013.120		Board Layout, Bds. 2-7
B13013.910		SCHEMATIC, Bd.2, ADC, DAC, REF, and Logic
B13014.000	PCB 3	ASSEMBLY DWG
B13014.010		ARTWORK LAYER 1
B13014.011		CLEARANCE LAYER FOR MARRIAGE WITH LAYER 1
B13014.020		ARTWORK LAYER 2
B13014.030		3
B13014.040		4
B13014.050		5
B13014.060		6
B13014.100		DRILL DWG
B13014.101		DRILL TABLE
B13014.110		SILKSCREEN
B13014.910		Bd.3, Protection Network
B13014.920		Bd.3, Pwr.Supply, Volt.Monitor
B13015.000	PDB 4	ASSEMBLY DWG
B13015.010		ARTWORK LAYER 1 (COMP. SIDE)
B13015.020		2
B13015.030		3
B13015.040		4
B13015.050		5
B13015.060		6 (SOLDER SIDE)
B13015.100		DRILL DWG
B13015.101		DRILL TABLE
B13015.110		SILKSCREEN
B13015.910		SCHEMATIC, PCSG, Waveform Select Reg.
B13015.920		PCSG, Timer
B13015.930		PCSG, Waveform Storage RAM
B13015.940		PCSG, Interrupt Control
B13015.950		PCSG, Edge Connector

Electronics Schematics and PCB Layout

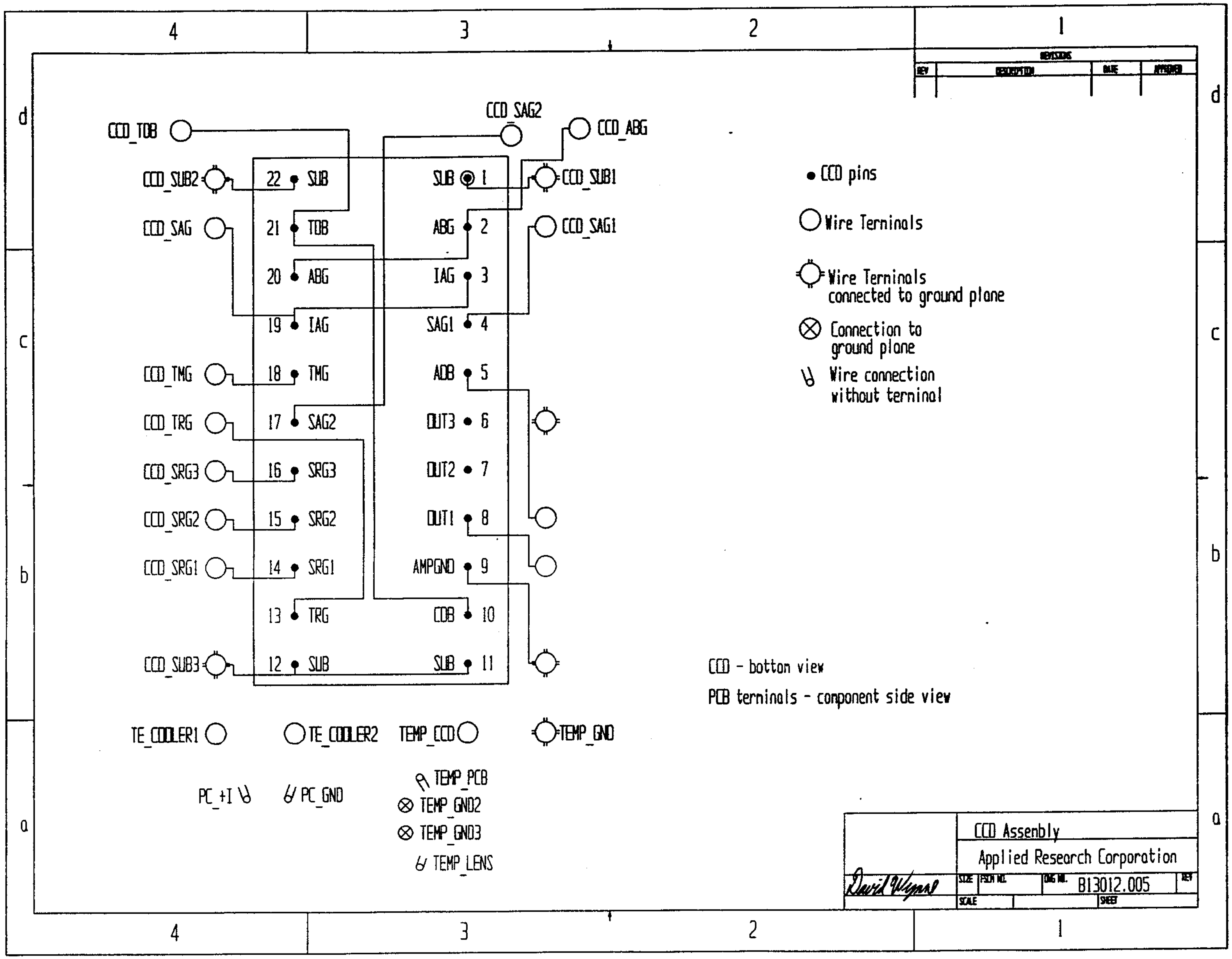
B13016.000	PCB 5	ASSEMBLY DWG
B13016.010		ARTWORK LAYER 1
B13016.020		2
B13016.030		3
B13016.040		4
B13016.050		5
B13016.060		6
B13016.100		DRILL DWG
B13016.101		DRILL TABLE
B13016.110		SILKSCREEN
B13016.900		SCHEMATIC, Bds. 5-7
B13016.910		Bd.5, CPU
B13017.000	PCB 6	ASSEMBLY DWG
B13017.010		ARTWORK LAYER 1
B13017.020		2
B13017.030		3
B13017.040		4
B13017.100		DRILL DWG
B13017.101		DRILL TABLE
B13017.110		SILKSCREEN
B13017.910		SCHEMATIC, Bd.6, Memory
B13018.000	PCB 7	ASSEMBLY DWG
B13018.010		ARTWORK LAYER 1
B13018.020		2
B13018.030		3
B13018.040		4
B13018.100		DRILL DWG
B13018.101		DRILL TABLE
B13018.110		SILKSCREEN
B13018.910		SCHEMATIC, Bd. 7, Interface

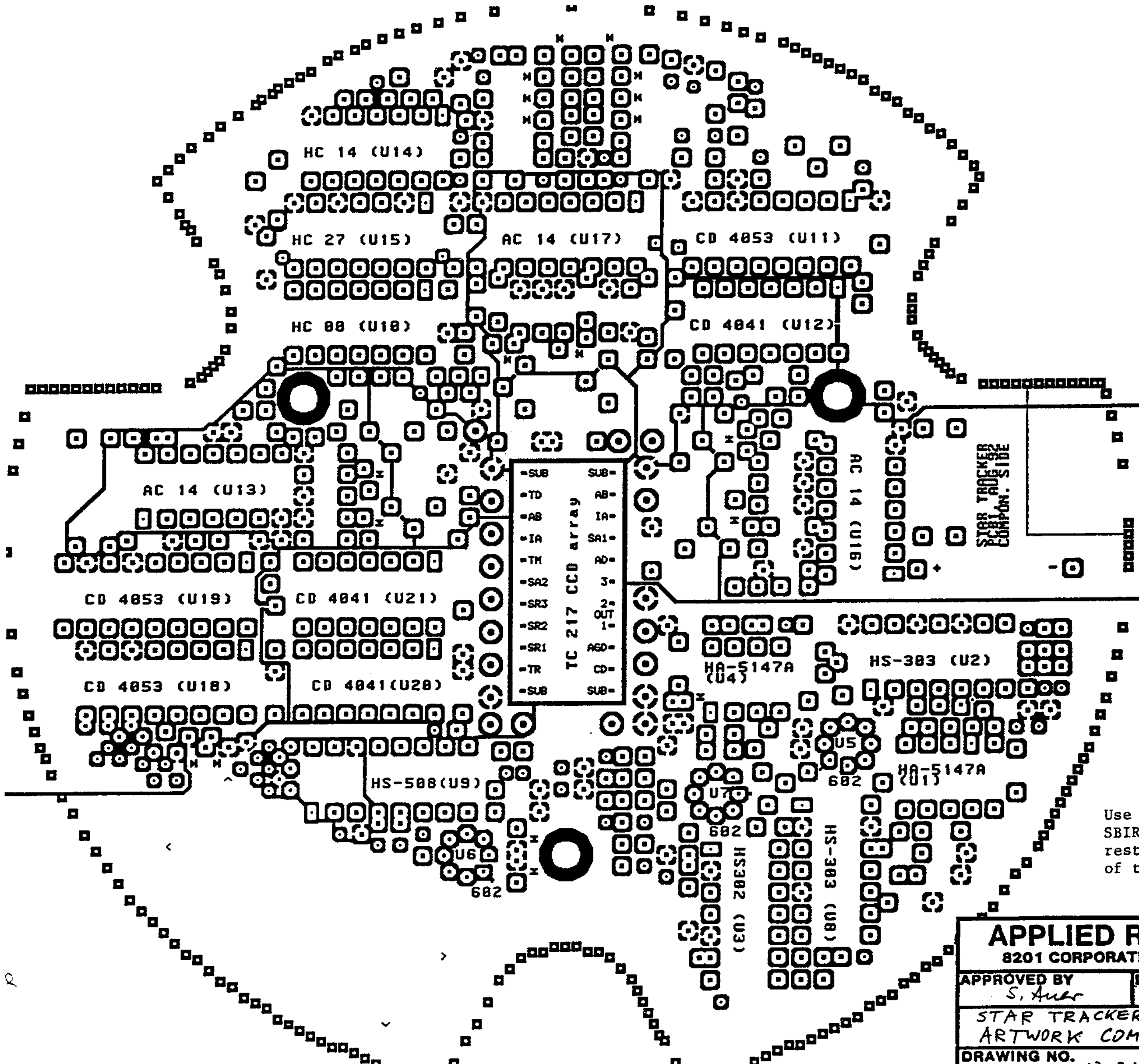


Approved but not signed
by Dr. S. Auer

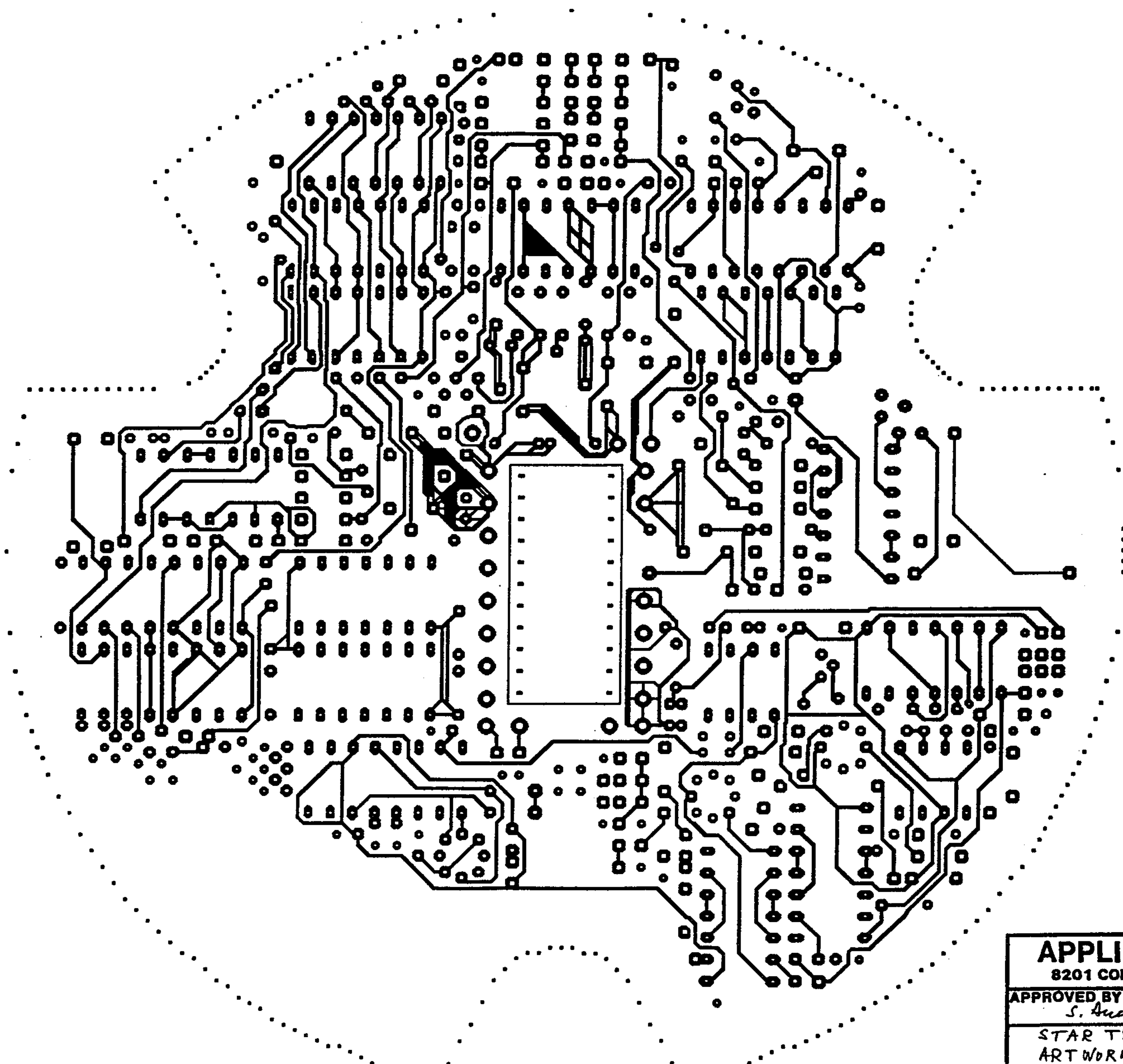
B13012.000

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.


CCD_TOB
CCD_SAG2
CCD_ABG
CCD_SUB2
CCD_SUB1
CCD_SAG
CCD_SAG1
CCD_TMG
CCD_TRG
CCD_SRG3
CCD_SRG2
CCD_SRG1
CCD_SUB3
22 SUB
21 TOB
20 ABG
19 IAG
18 TMG
17 SAG2
16 SRG3
15 SRG2
14 SRG1
13 TRG
12 SUB
SUB 1
ABG 2
IAG 3
SAG1 4
ADB 5
OUT3 6
OUT2 7
OUT1 8
AMPGND 9
CDB 10
SUB 11
TE_COOLER1
TE_COOLER2
TEMP_CCD
TEMP_GND
PC_+I
PC_GND
TEMP_PCB
TEMP_GND2
TEMP_GND3
TEMP_LENS
CCD pins
Wire Terminals
Wire Terminals connected to ground plane
Connection to ground plane
Wire connection without terminal
CCD - botton view
PCB terminals - component side view
CCD Assenbly
Applied Research Corporation
B13012.005

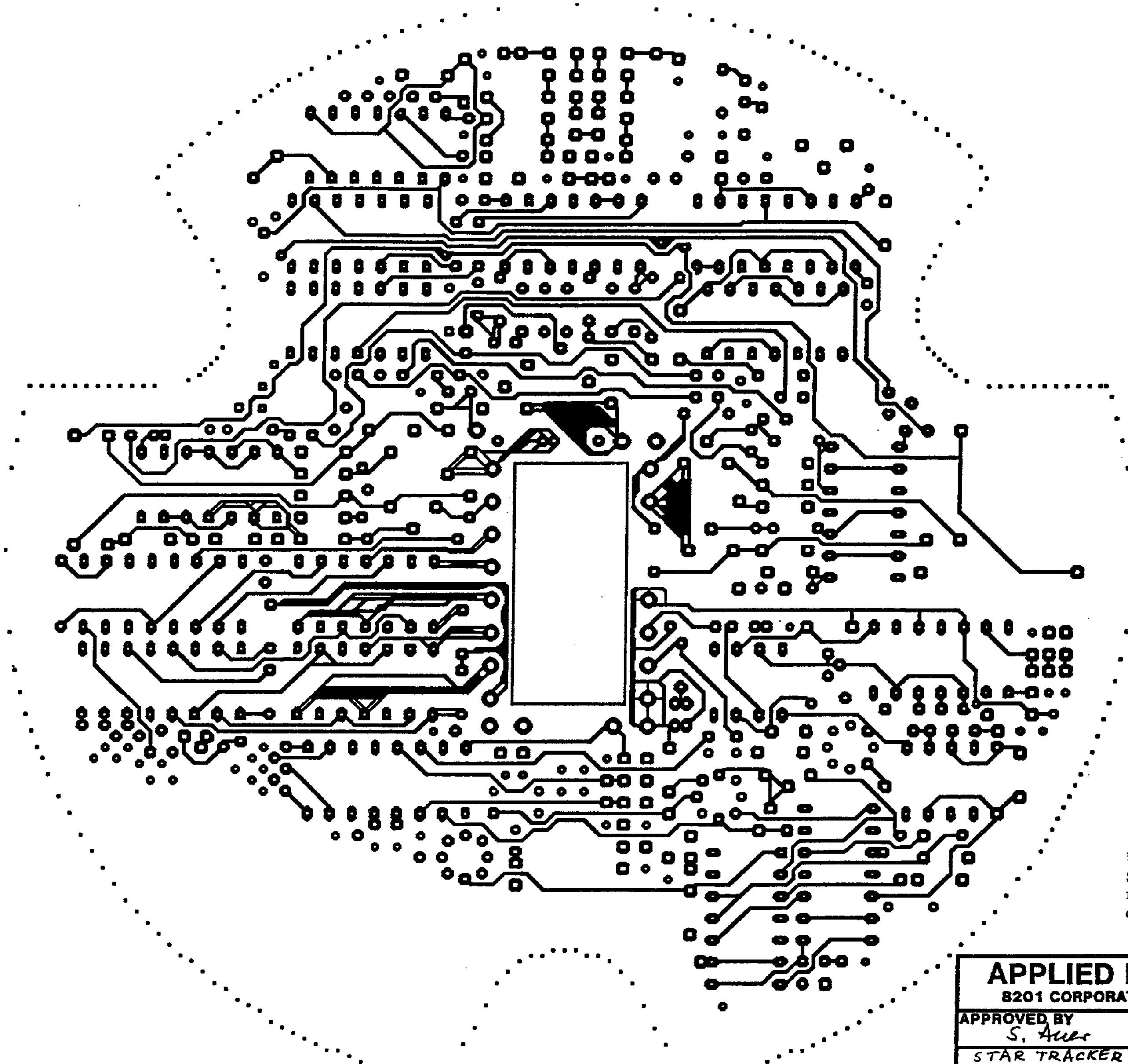


APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY S. Auer	DRAWN BY S. AUER	DATE
STAR TRACKER, PCB 1 ARTWORK COMPONENT SIDE (LAYER 1)		
DRAWING NO. 13012.010		SHEET NO.



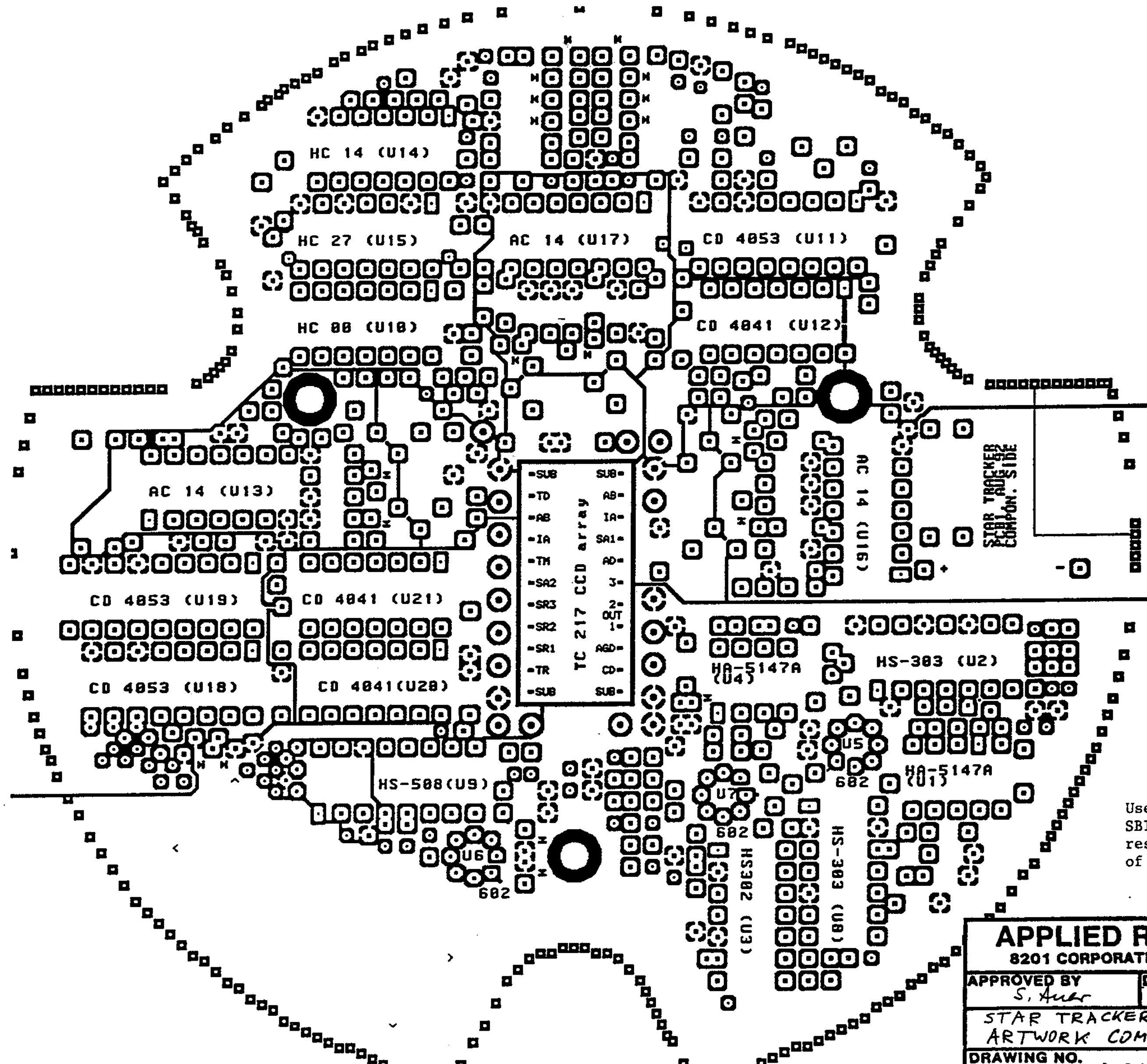
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY S. Auer	DRAWN BY S. AUER	DATE 8-25-92
STAR TRACKER, PCB 1 ARTWORK, LAYER 2		
DRAWING NO. 13012.020		SHEET NO.

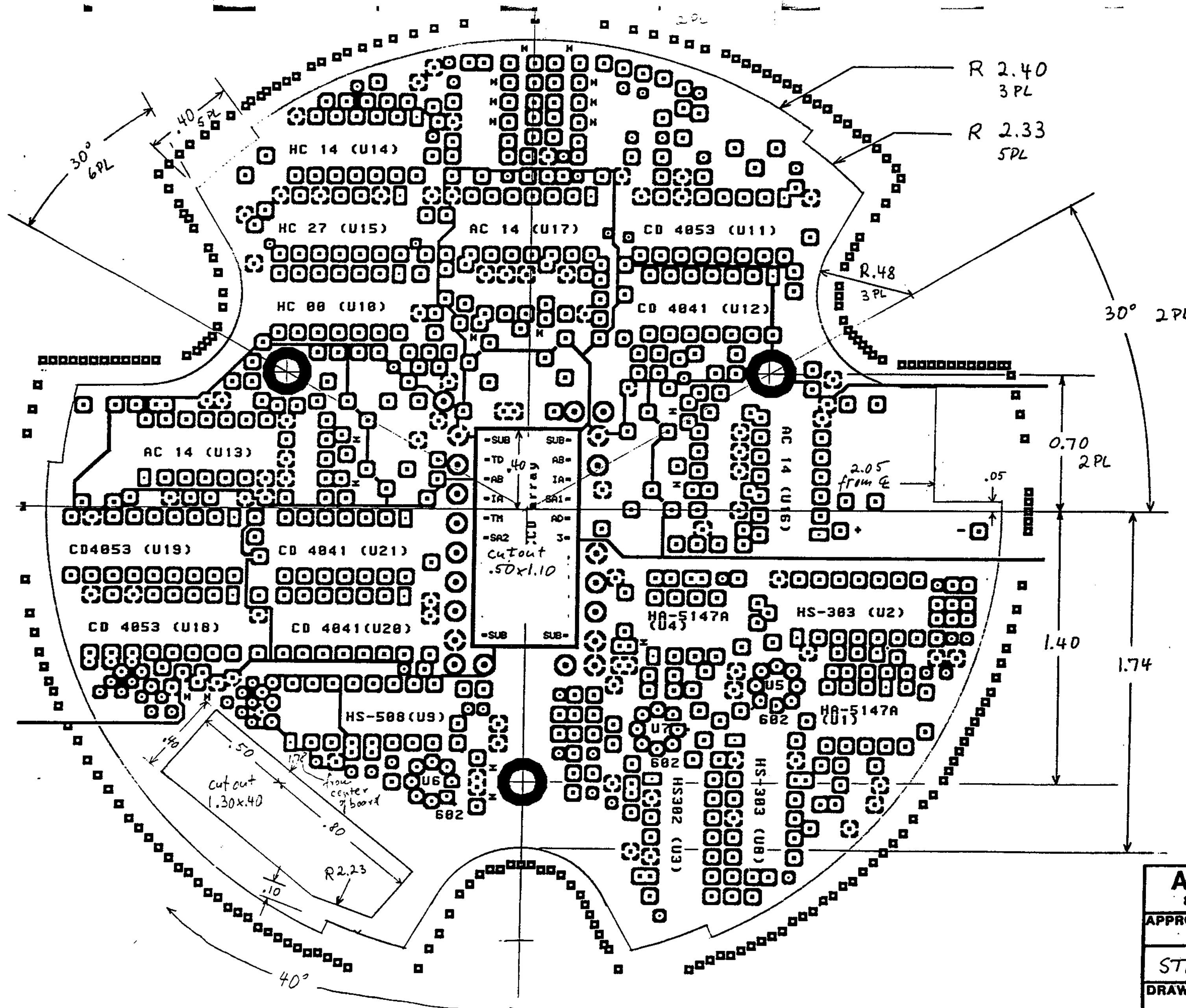


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY S. Auer	DRAWN BY S. AUER	DATE 8-25-92
STAR TRACKER, PCB 1 ARTWORK, LAYER 3		
DRAWING NO. 13012.030		SHEET NO.



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY S. Auer	DRAWN BY S. AUER	DATE
STAR TRACKER, PCB 4 ARTWORK COMPONENT SIDE (LAYER 1)		
DRAWING NO. 13012.040		SHEET NO.



HOLE DATA

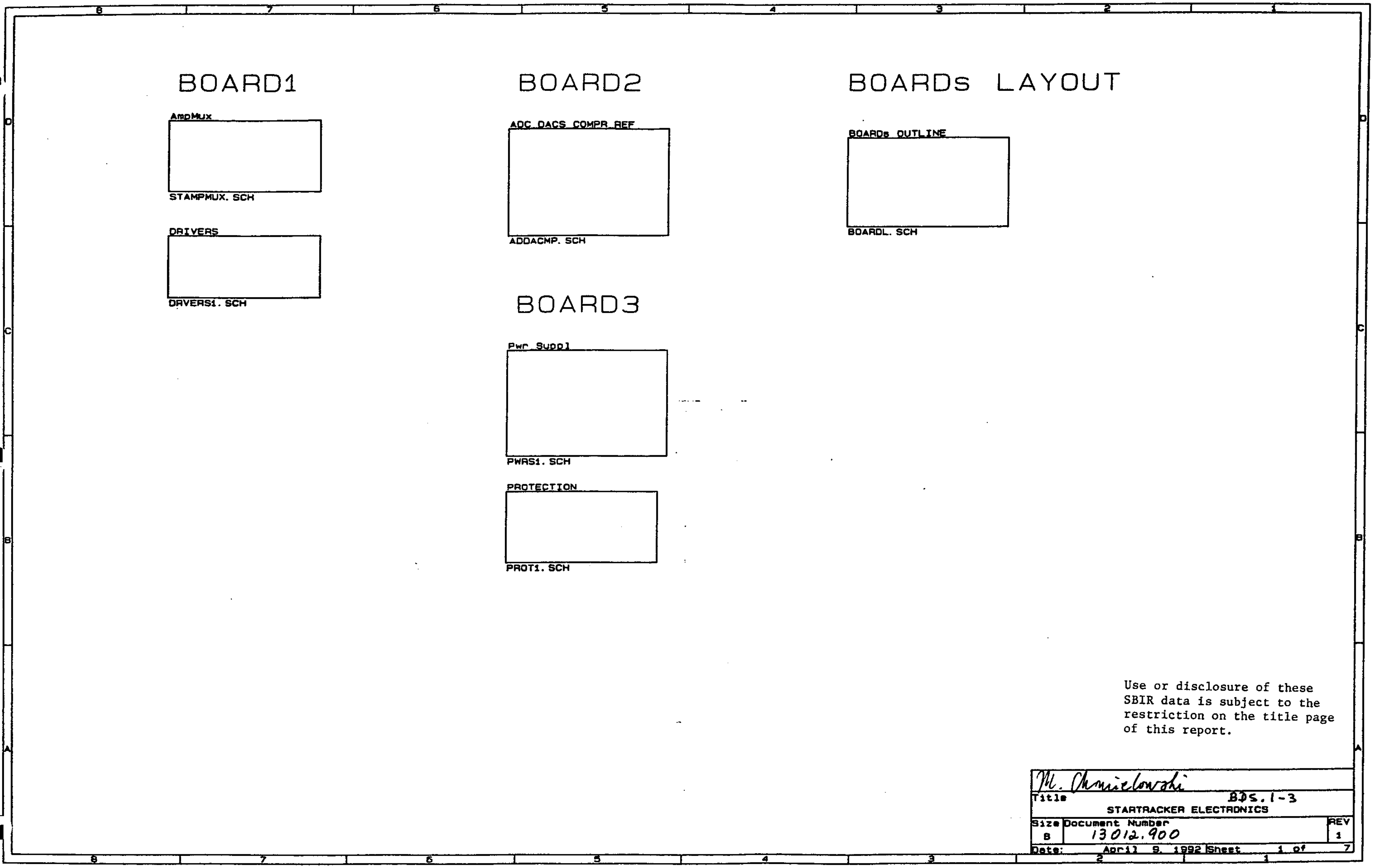
DESIGNATION	HOLE DIA.	QTY
UNMARKED	$.025^{+.003}_{-.001}$	-
A	$.030^{+.003}_{-.001}$	184
B	.020 ± .003	5
C	$.043^{+.003}_{-.001}$	52
D	$.064^{+.004}_{-.000}$	21
E	.155 ± .004	3

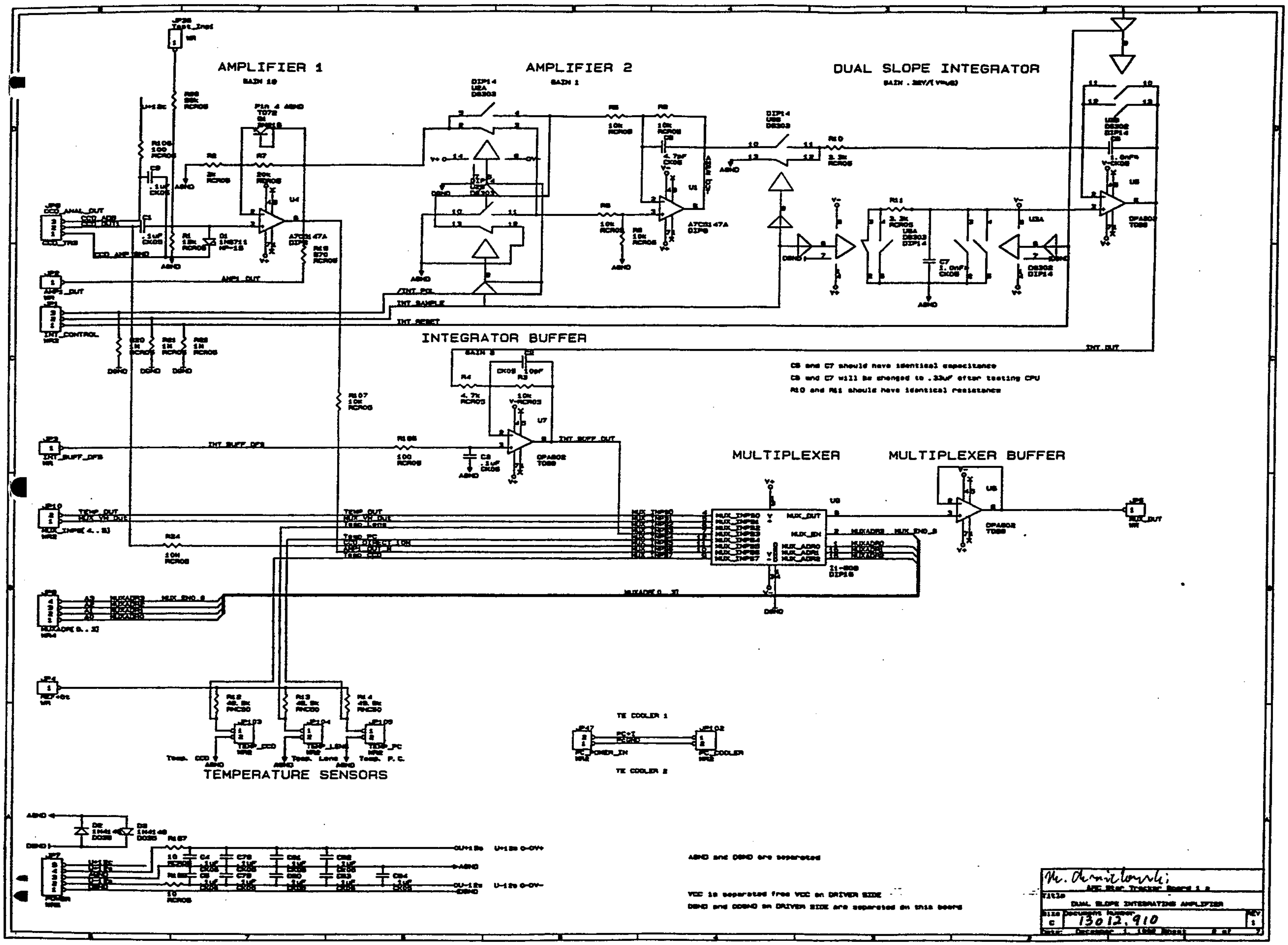
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

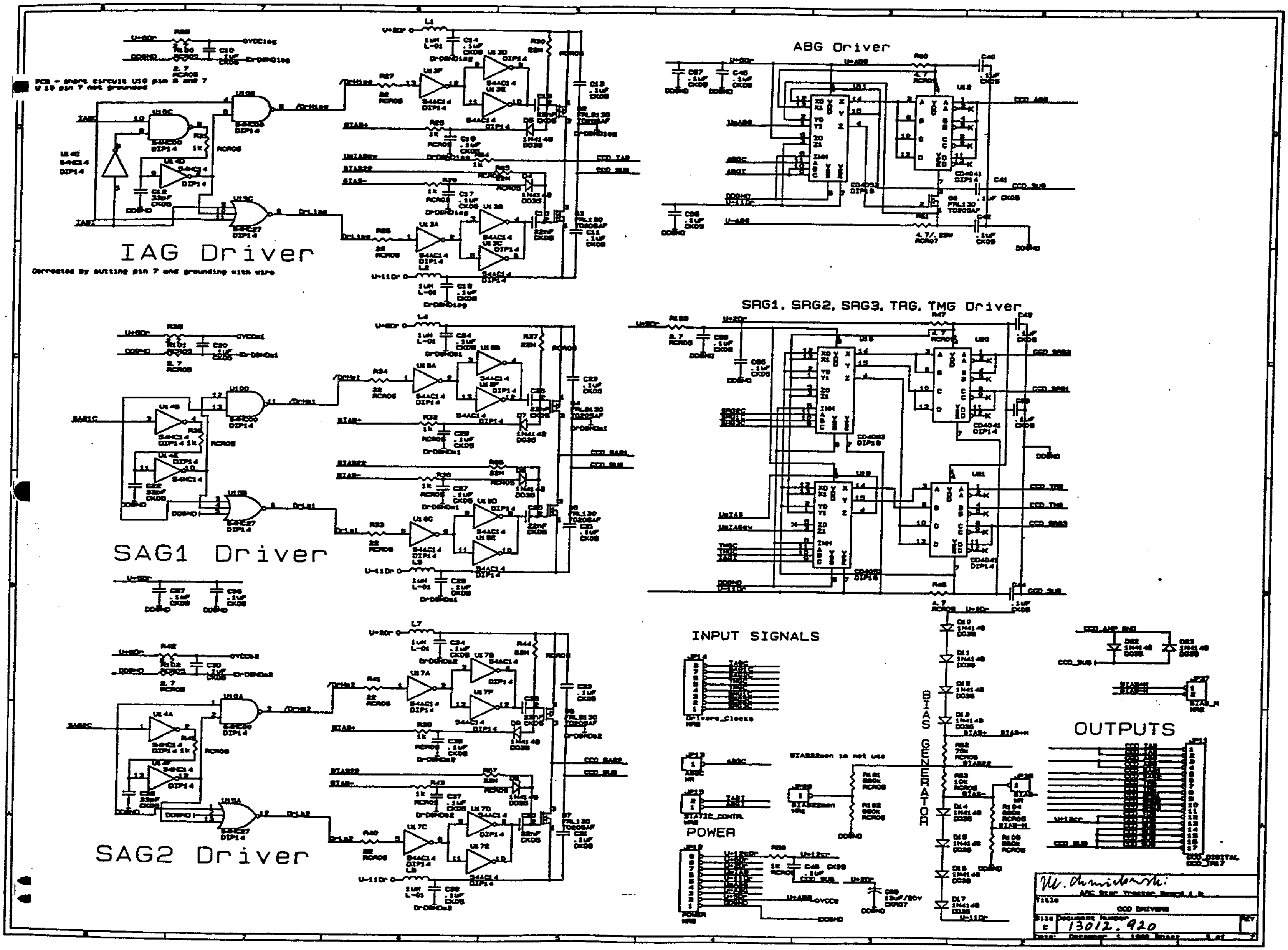
APPROVED BY	DRAWN BY	DATE
S. Auer	S. AUER	8-25-92

STAR TRACKER PCB 1 FABRICATION

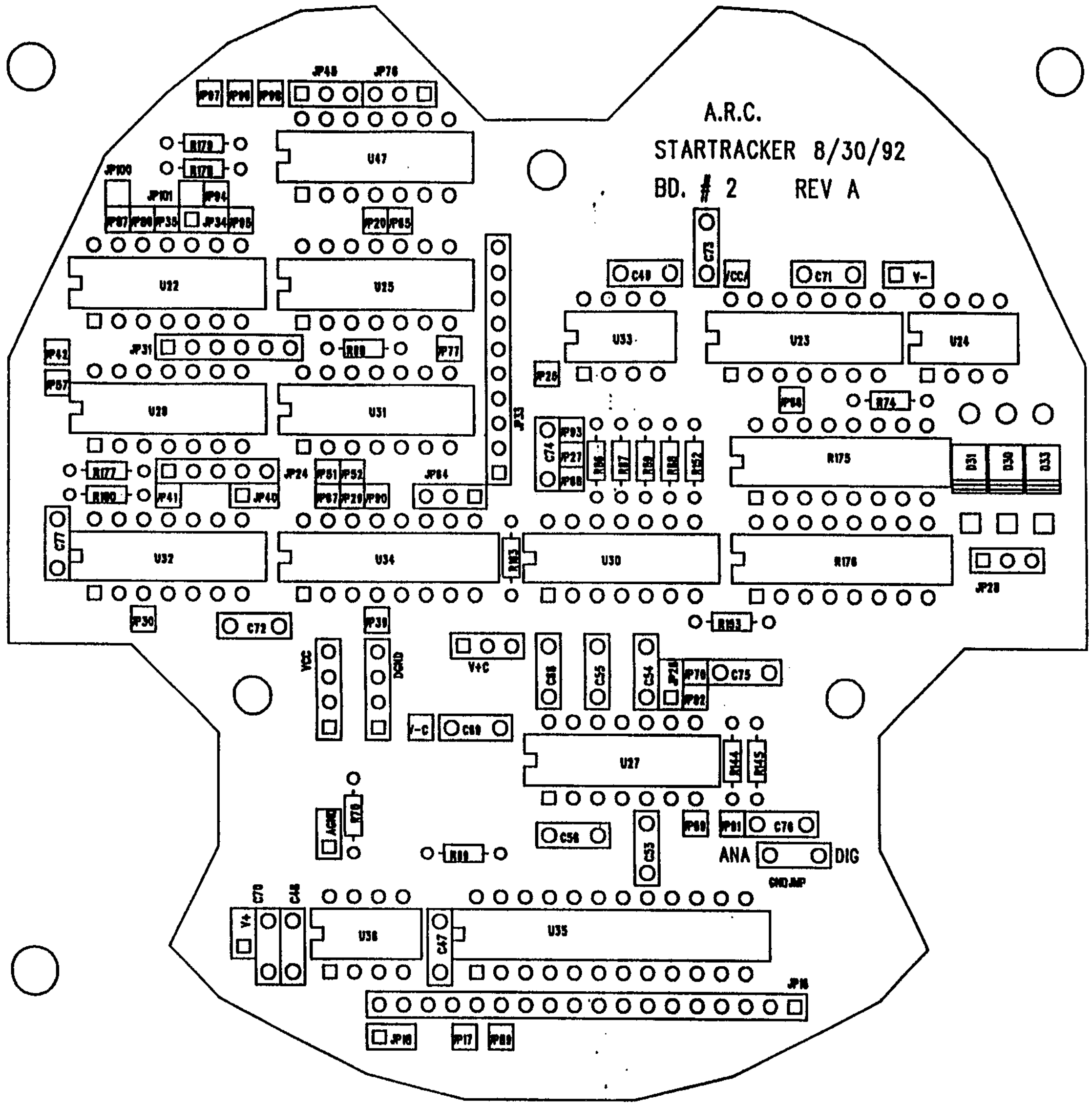
DRAWING NO.	SHEET NO.
13012.100	1 OF 2


BOARD1
AmpMux
STAMPMUX. SCH
DRIVERS
DRVERS1. SCH
BOARD2
ADC DACS COMPR REF
ADDACMP. SCH
BOARD3
Pwr Suppl
PWRS1. SCH
PROTECTION
PROT1. SCH
BOARDs LAYOUT
BOARDs OUTLINE
BOARDL. SCH
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.
M. Chmielowski
Title BDS. 1-3
STARTRACKER ELECTRONICS
Size B
Document Number 13012.900
REV 1
Date: April 9, 1992
Sheet 1 of 7


AMPLIFIER 1
AMPLIFIER 2
DUAL SLOPE INTEGRATOR
INTEGRATOR BUFFER
MULTIPLEXER
MULTIPLEXER BUFFER
TEMPERATURE SENSORS
TE COOLER 1
TE COOLER 2
DUAL SLOPE INTEGRATING AMPLIFIER
13012.910


IAG Driver
ABG Driver
SRG1, SRG2, SRG3, TRG, TMG Driver
SAG1 Driver
SAG2 Driver
INPUT SIGNALS
POWER
BIAS GENERATOR
OUTPUTS
CCD DRIVERS
13012.920

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92
ASSEMBLY DRAWING



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

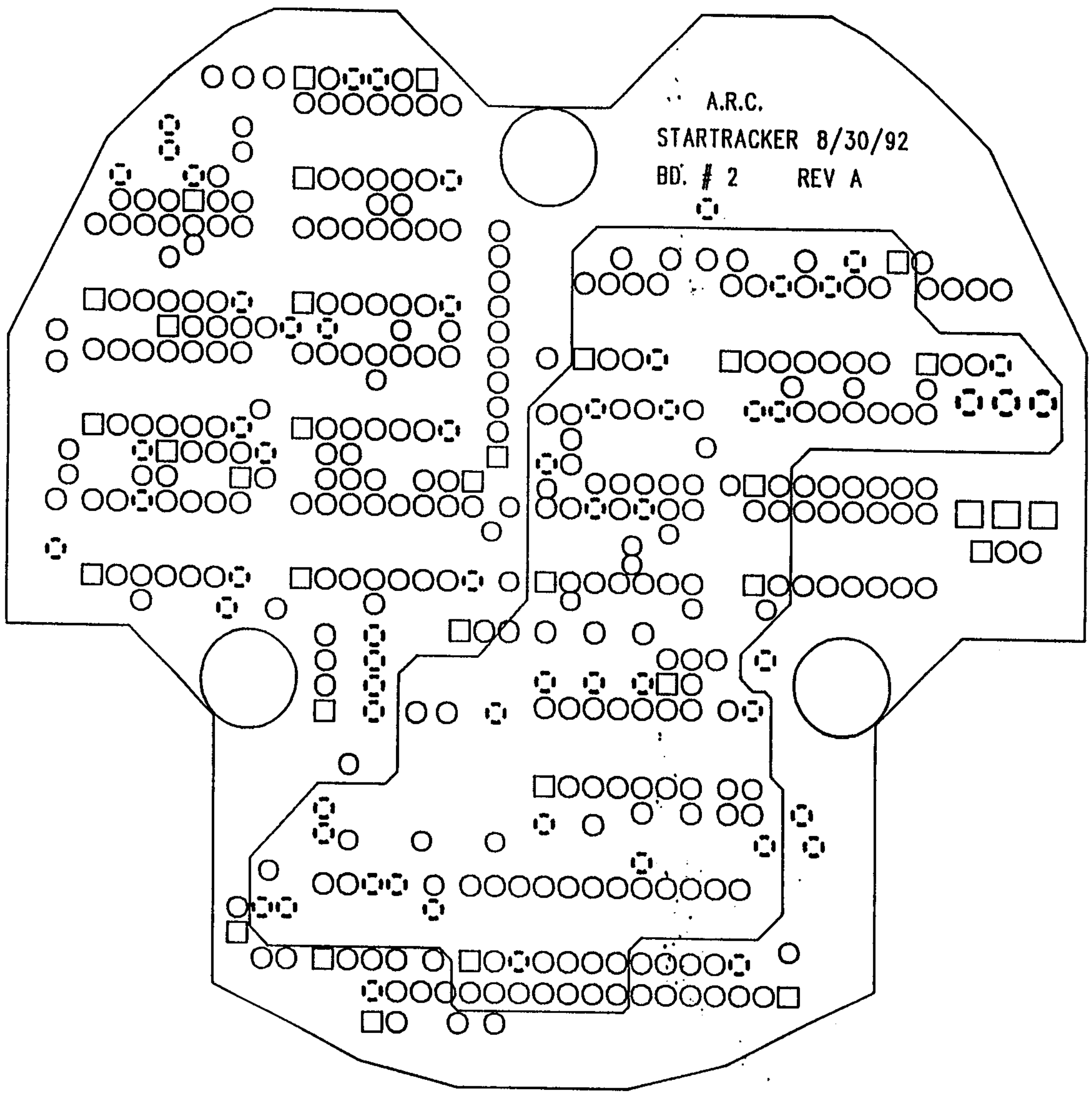
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
B 13013.000	

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92
GROUND PLANE
LAYER 1 OF 4

CLEARANCE LAYER
FOR MARRIAGE WITH LAYER 1



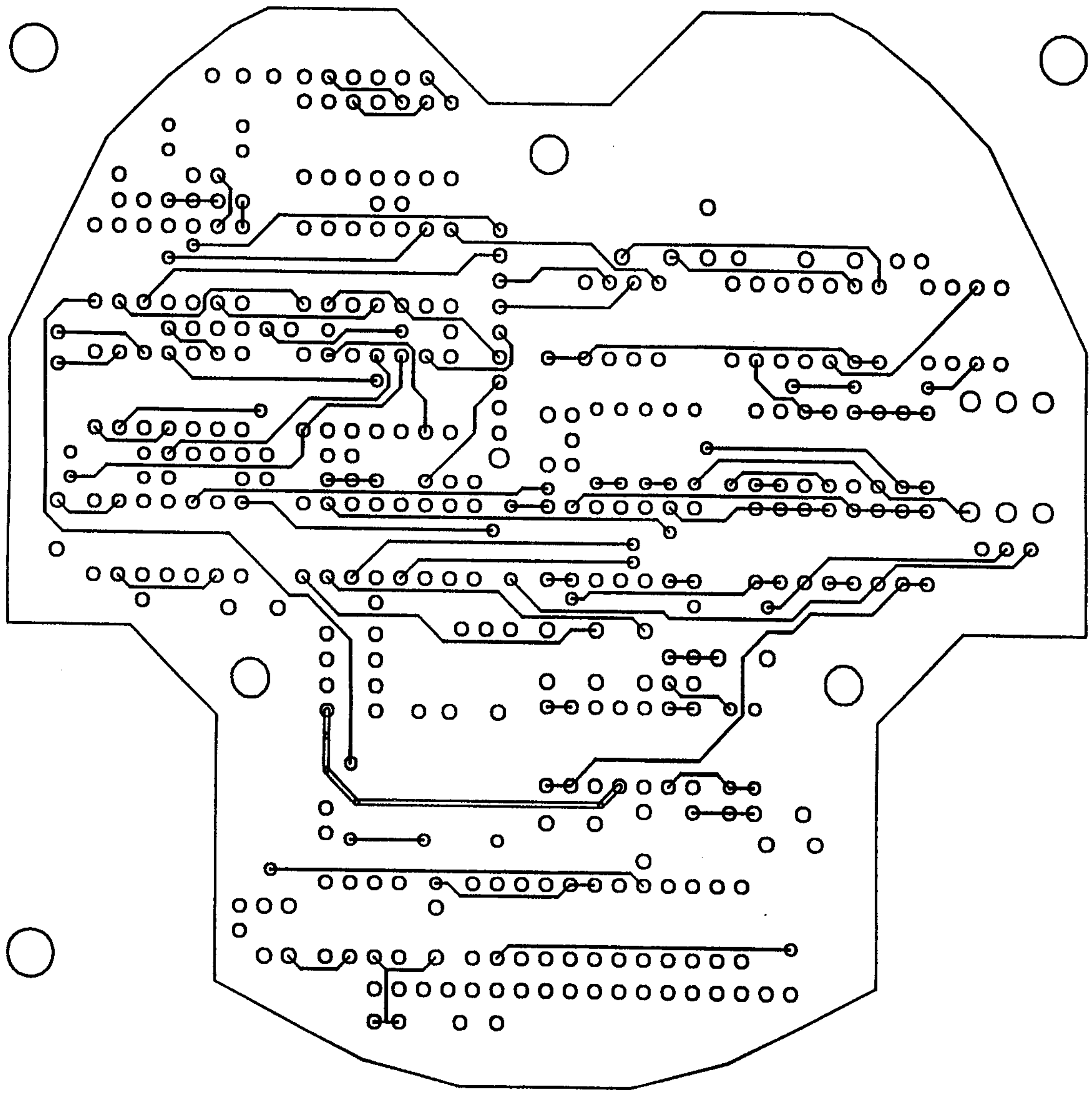
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13013.010		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92

SIGNAL LAYER

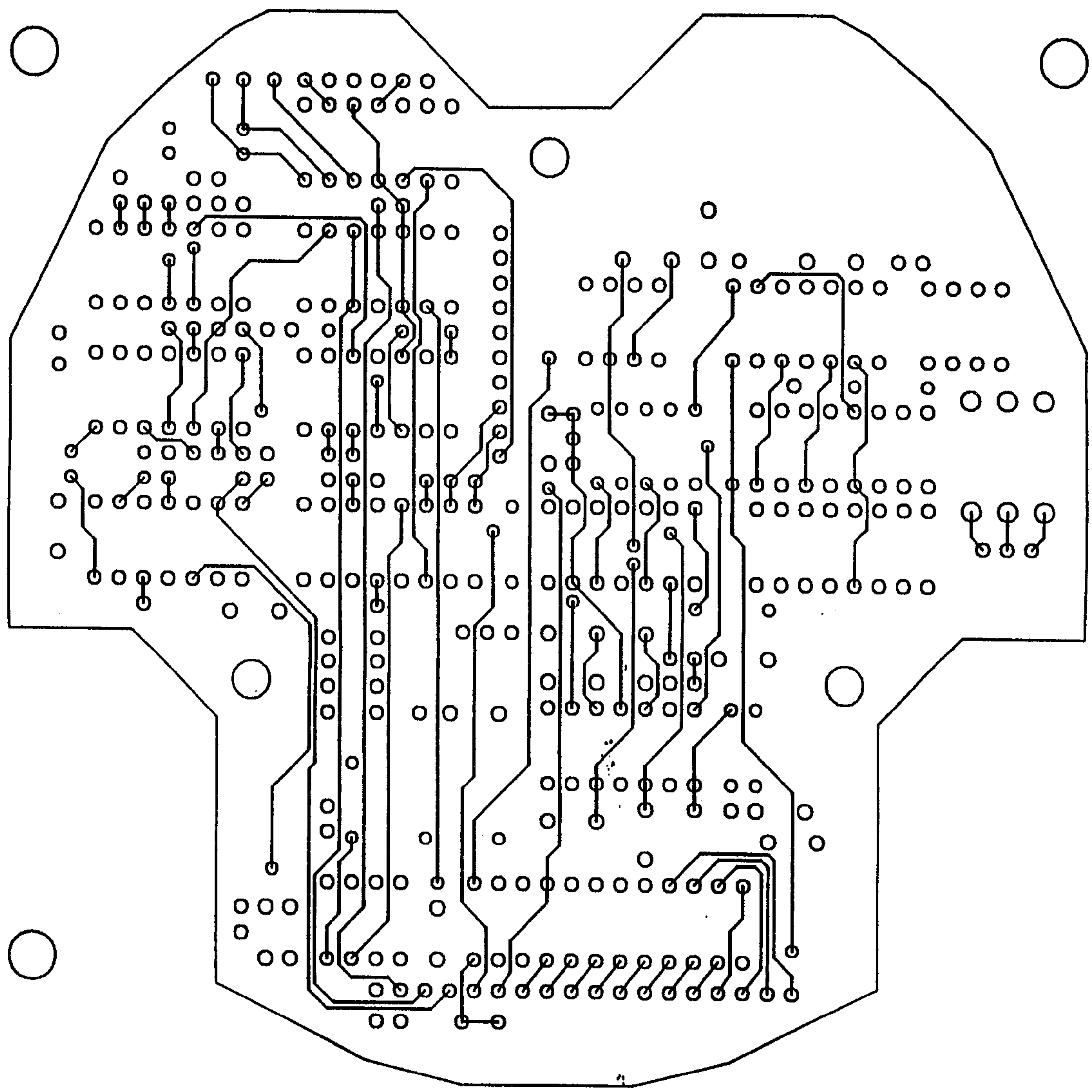
LAYER 2 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13013.020		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92
SIGNAL LAYER
LAYER 3 OF 4



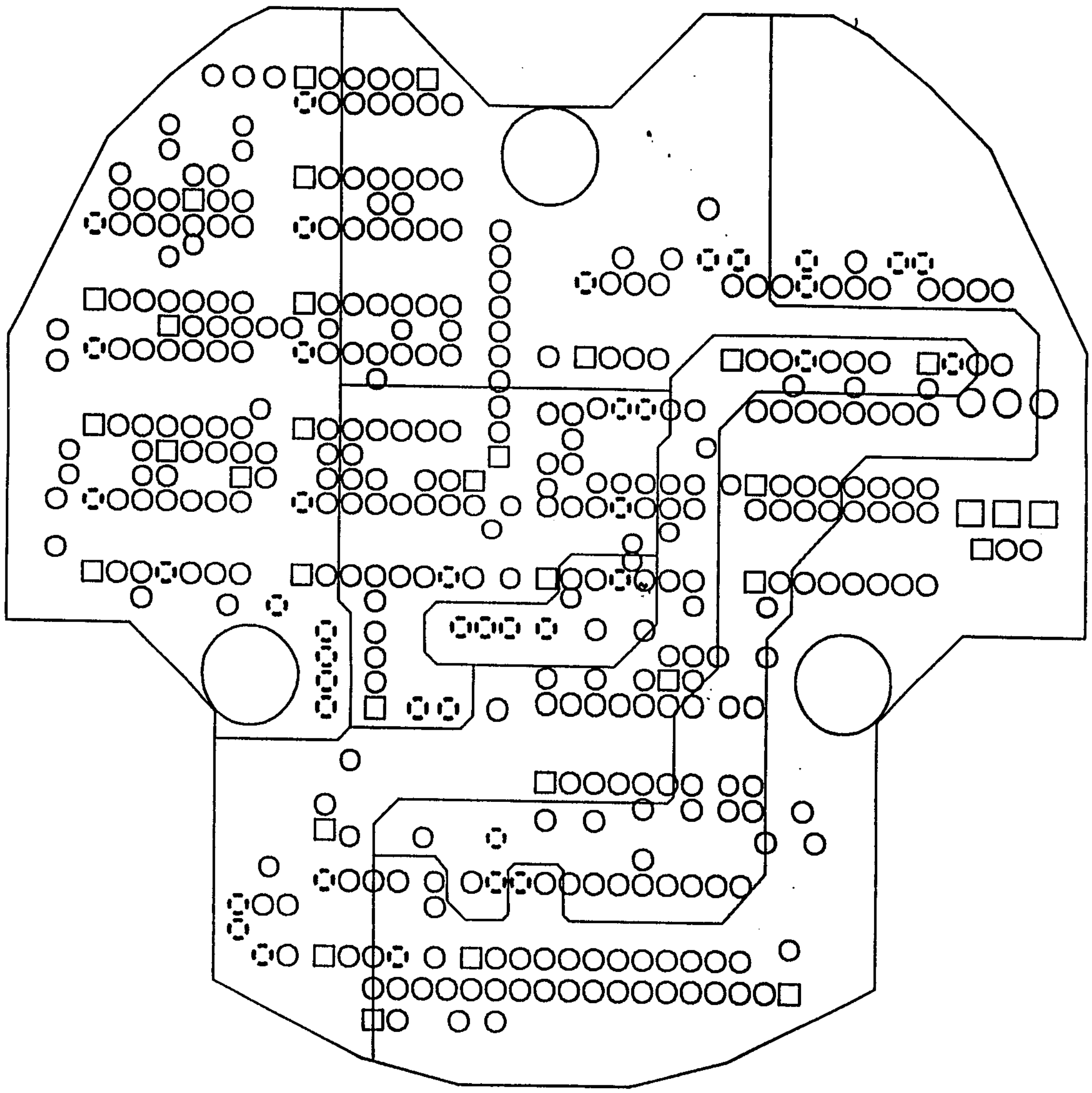
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	
DRAWING NO. B13013.030		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92
POWER PLANE
LAYER 4 OF 4

CLEARANCE LAYER
FOR MARRIAGE WITH LAYER 4



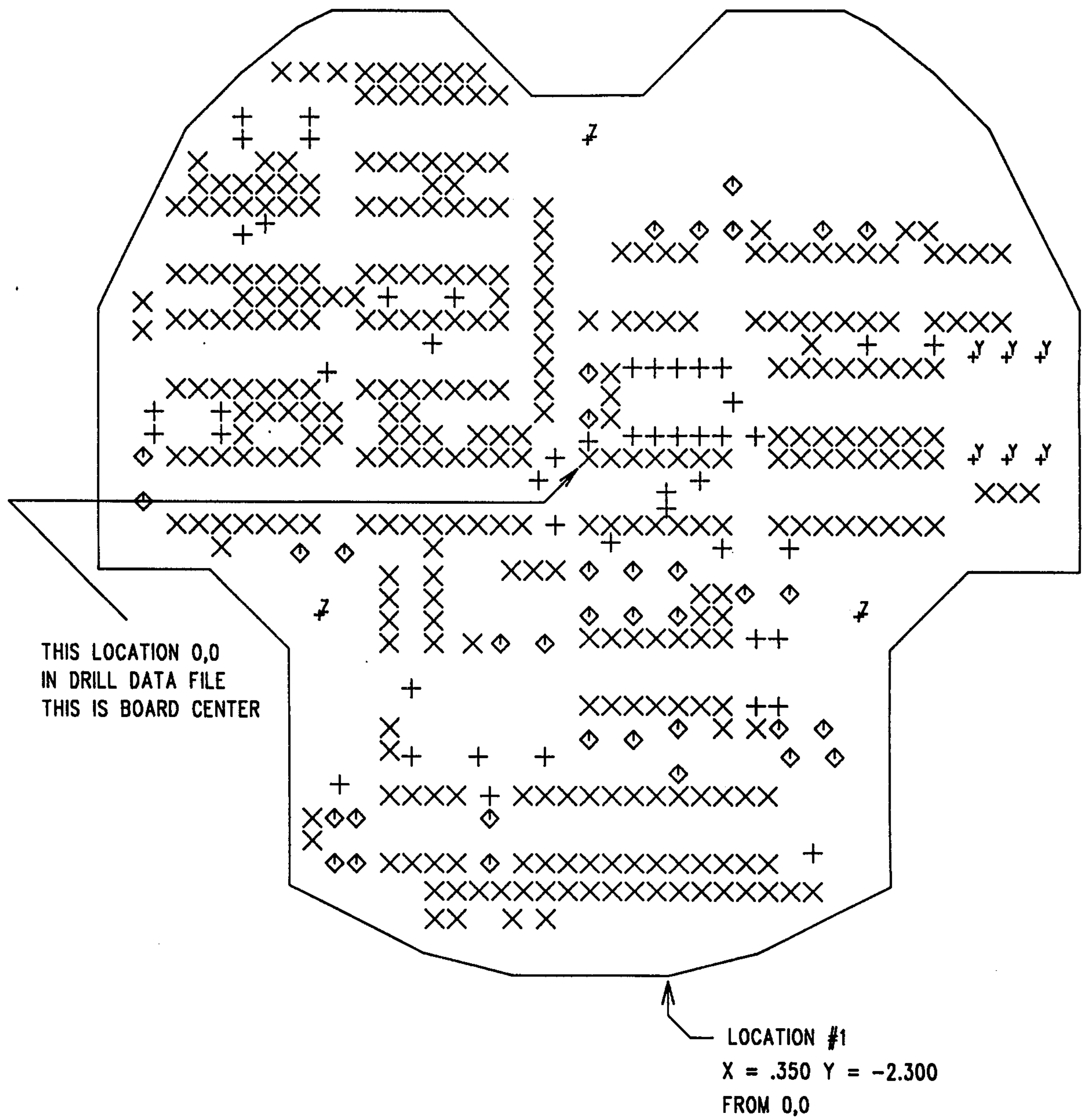
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
B13013.040	

APPLIED RESEARCH CORPORATION DRILL DRAWING

STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92



Approved but not signed
by Dr. S. Auer

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY	DRAWN BY PHIL GOODWIN	DATE 8/30/92
DRILL DRAWING		
DRAWING NO. B13013.100		SHEET NO.

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

SIZE	QTY	SYM
28	42	+
32	463	X
37	40	◇
160	3	Z

BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION #1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.700	1.450
6.	-2.050	1.800
7.	-2.550	1.800
8.	-2.550	2.950
9.	-2.150	3.750
10.	-1.900	4.000
11.	-1.650	4.200
12.	-1.500	4.275
13.	-.975	4.275
14.	-.600	3.900
15.	-.100	3.900
16.	.275	4.275
17.	.800	4.275
18.	.950	4.200
19.	1.200	4.000
20.	1.450	3.750
21.	1.850	2.950
22.	1.850	1.800
23.	1.350	1.800
24.	1.000	1.450
25.	1.000	.400
26.	.400	.100
27.	0	0

FABRICATION NOTES:

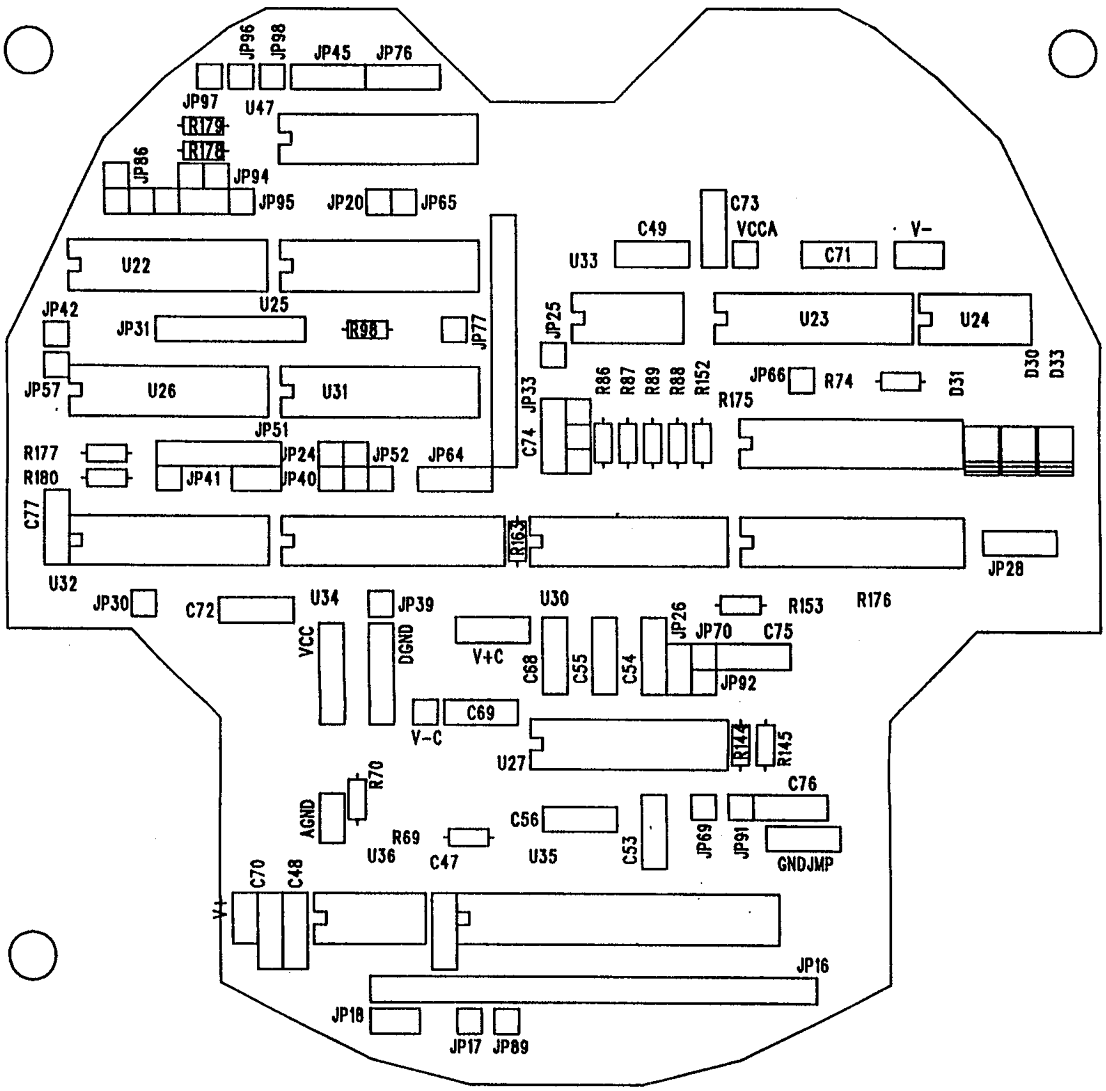
FINISHED BOARD THICKNESS TO BE .062 +/- .007
MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
TYPE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
FABRICATE IN ACCORDANCE WITH MIL-P-55110
ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
LAYER 1 -COMPONENT SIDE THRU LAYER 4 -SOLDER SIDE
MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
PER MIL-I-43553
ALL .160" DIA. HOLES WILL BE NON-PLATED THRU (3PLCS.)

Approved but not signed
by Dr. S. Auer

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY	DRAWN BY PHIL GOODWIN	DATE 8/30/92
DRILL TABLE		
DRAWING NO. B13013,101		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #2 REV A 8/30/92

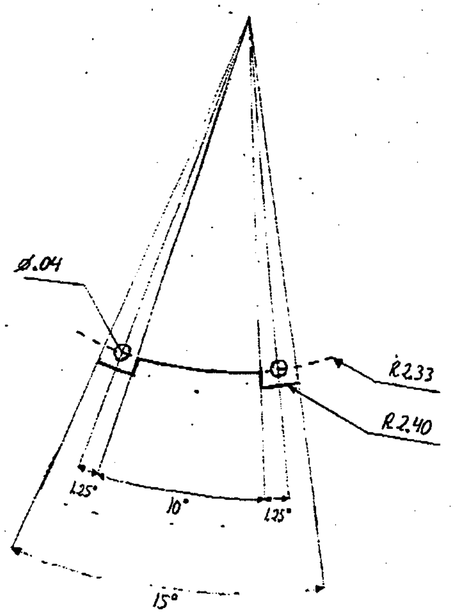
SILKSCREEN
LAYER 1 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

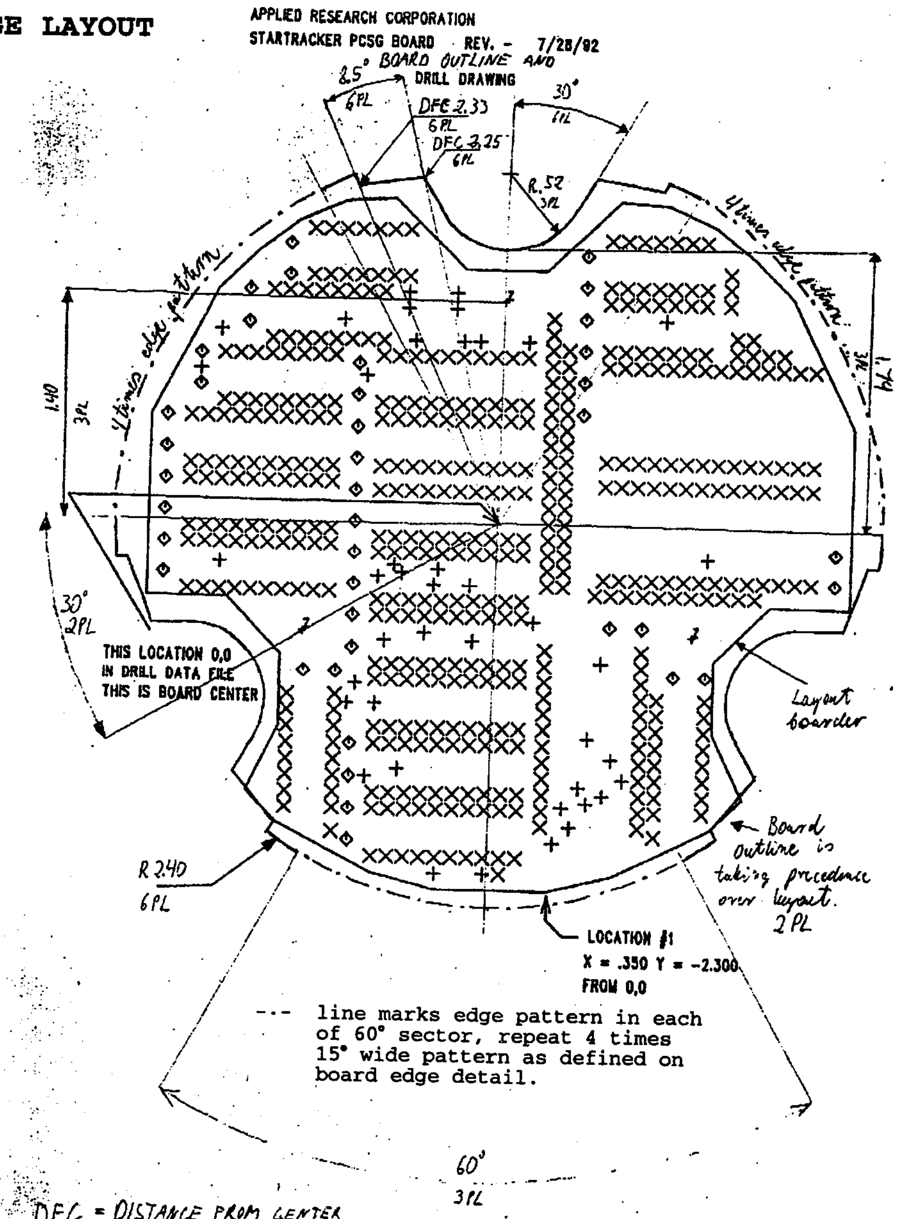
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13013.110		SHEET NO.

BOARD EDGE LAYOUT



Board edge pattern should be repeated in each of the 3 sectors - total 12 times.

Board Edge Detail



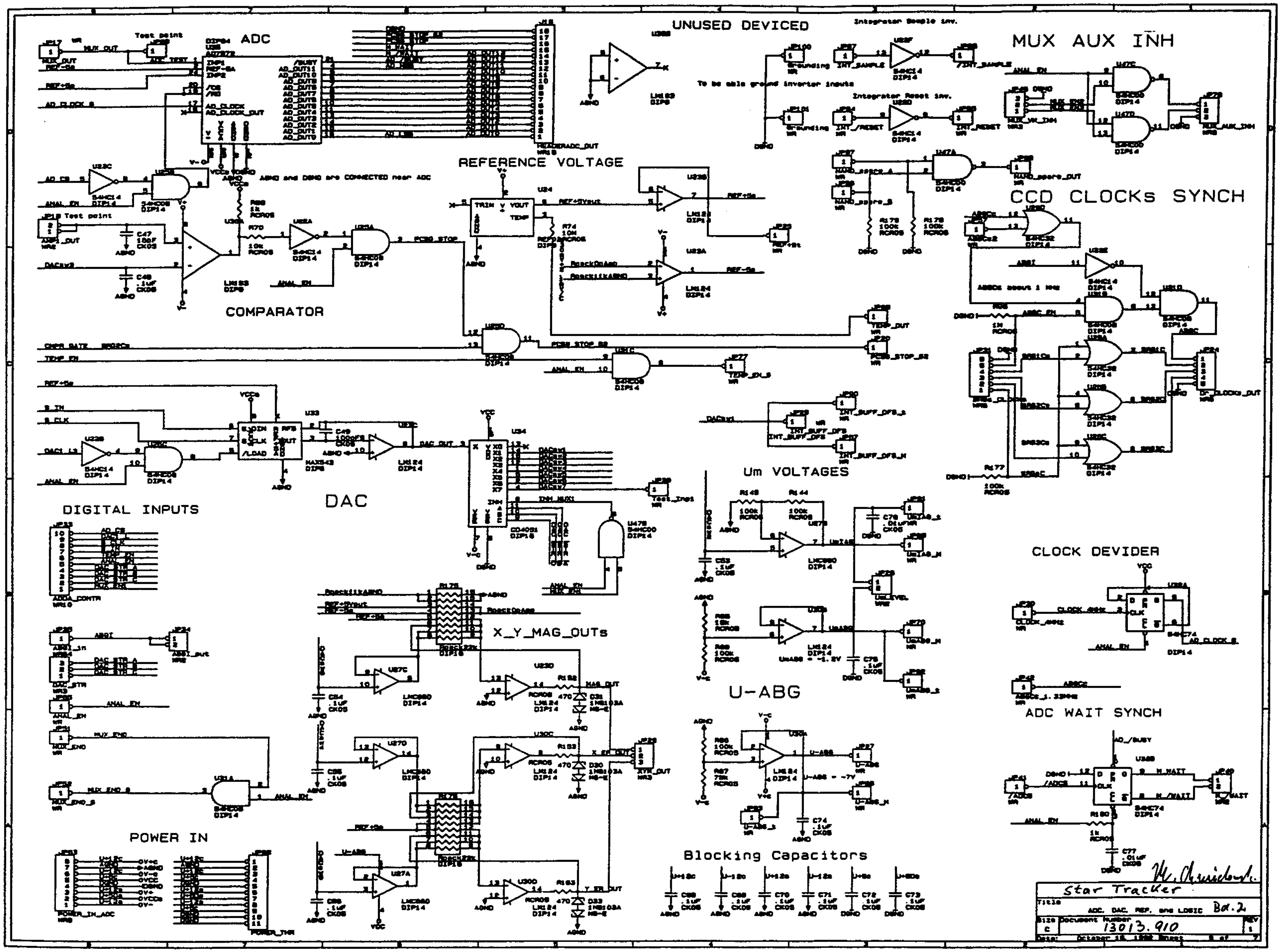
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
	M. Church	11/3/92

Board Edge Layout

DRAWING NO. B13013.120 | SHEET NO.

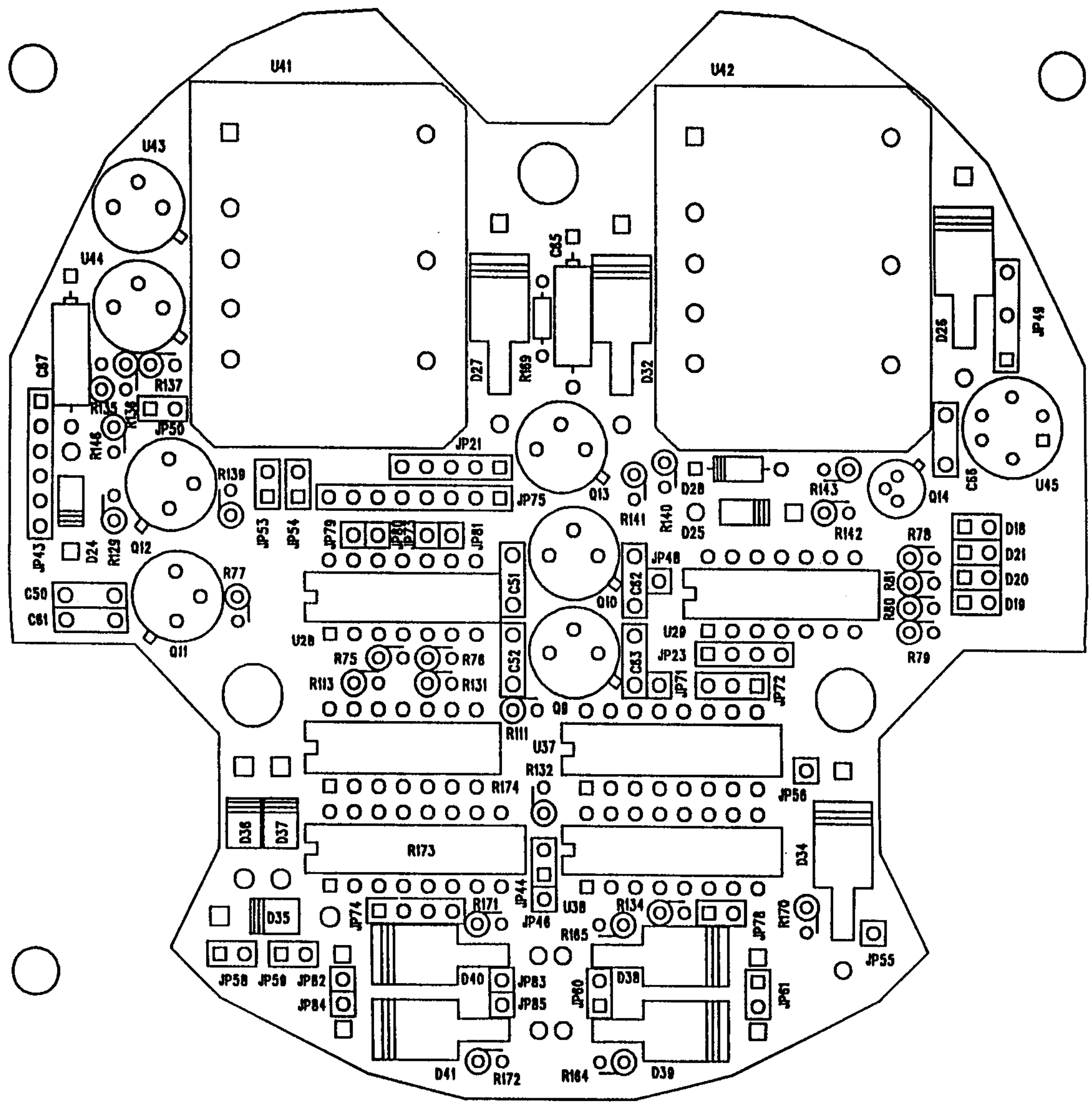
Approved but not signed by Dr. S.


ADC
UNUSED DEVICED
MUX AUX INH
REFERENCE VOLTAGE
CCD CLOCKs SYNCH
COMPARATOR
DIGITAL INPUTS
DAC
Um VOLTAGES
CLOCK DEVIDER
X_Y_MAG_OUTs
U-ABG
ADC WAIT SYNCH
POWER IN
Blocking Capacitors
Star Tracker
ADC, DAC, REF, and LOGIC
13013.910

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92

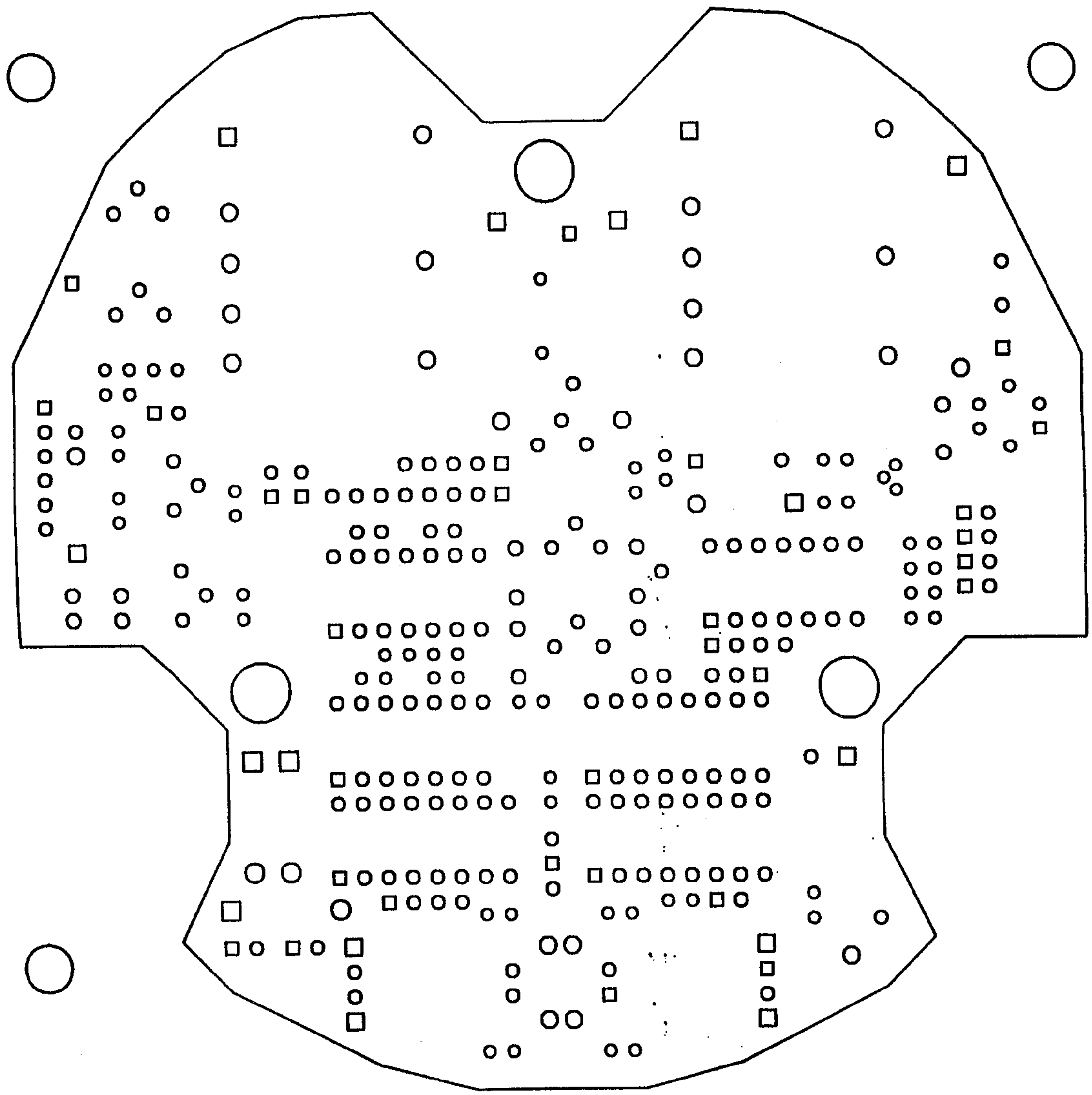
ASSEMBLY DRAWING



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13014.000		SHEET NO.

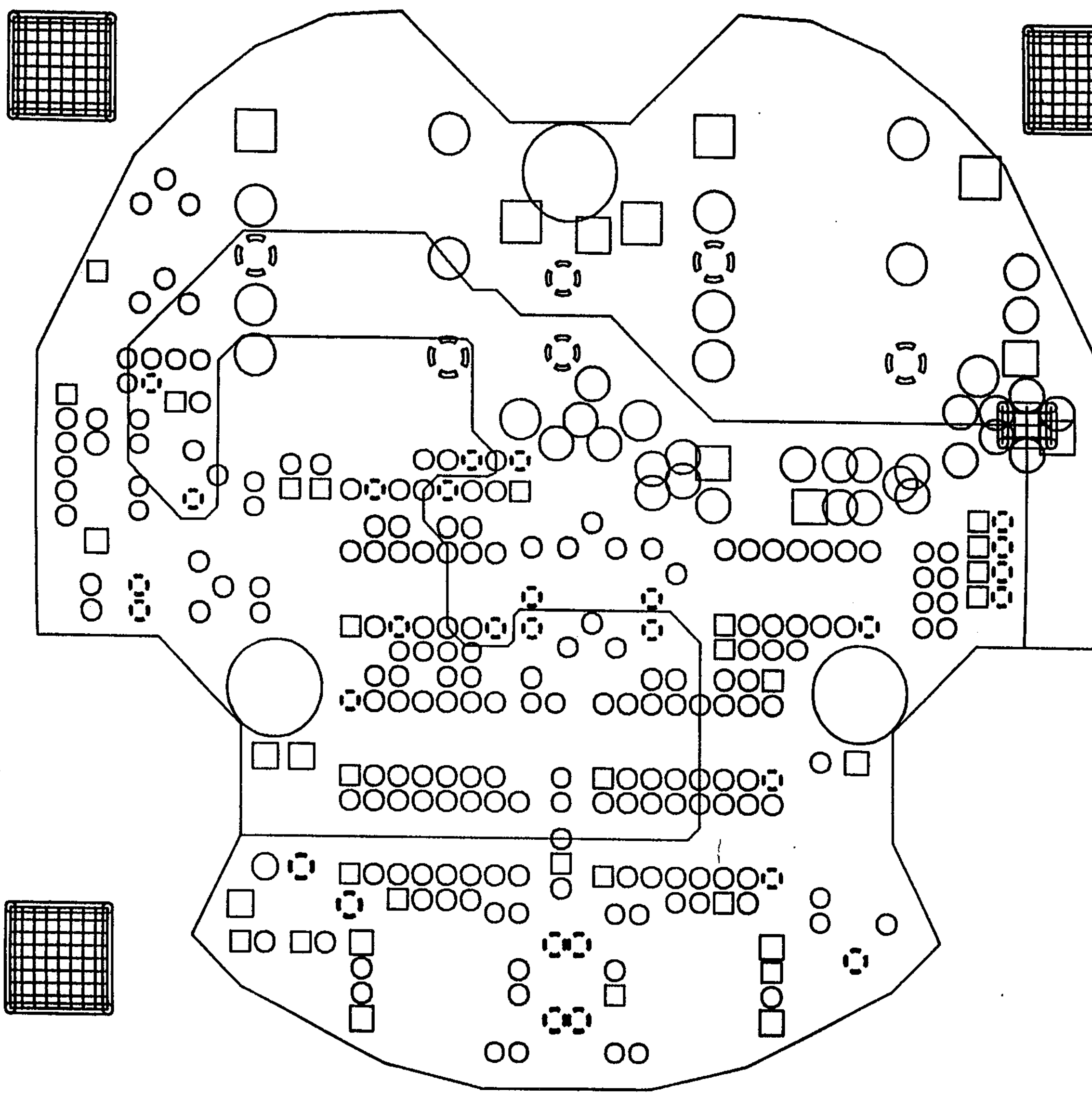
APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92
GROUND PLANE
LAYER 1 OF 6



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13014.010		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/97

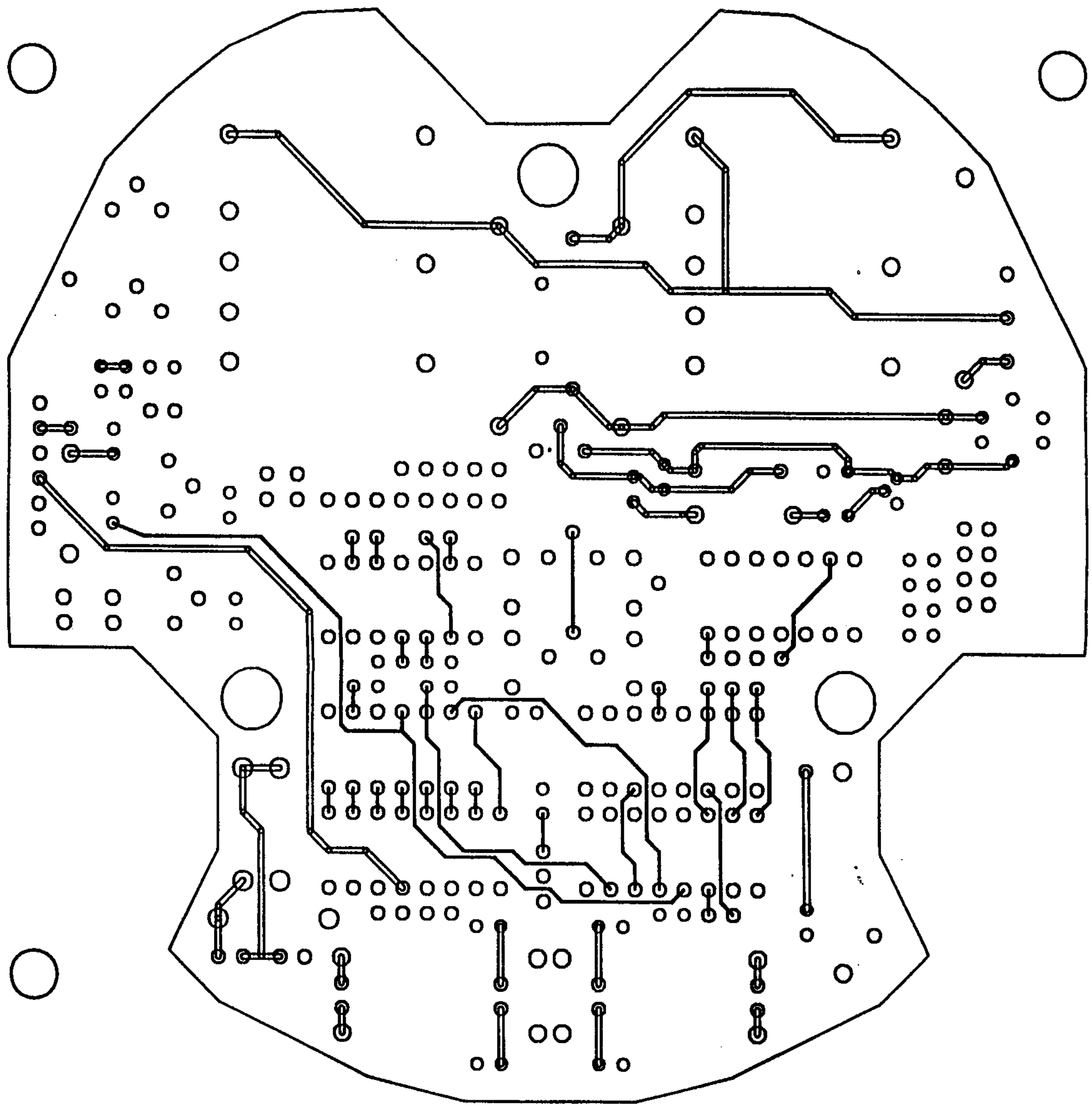
CLEARANCE LAYER
FOR MARRIAGE WITH LAYER



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13014.011		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92
SIGNAL LAYER
LAYER 2 OF 6



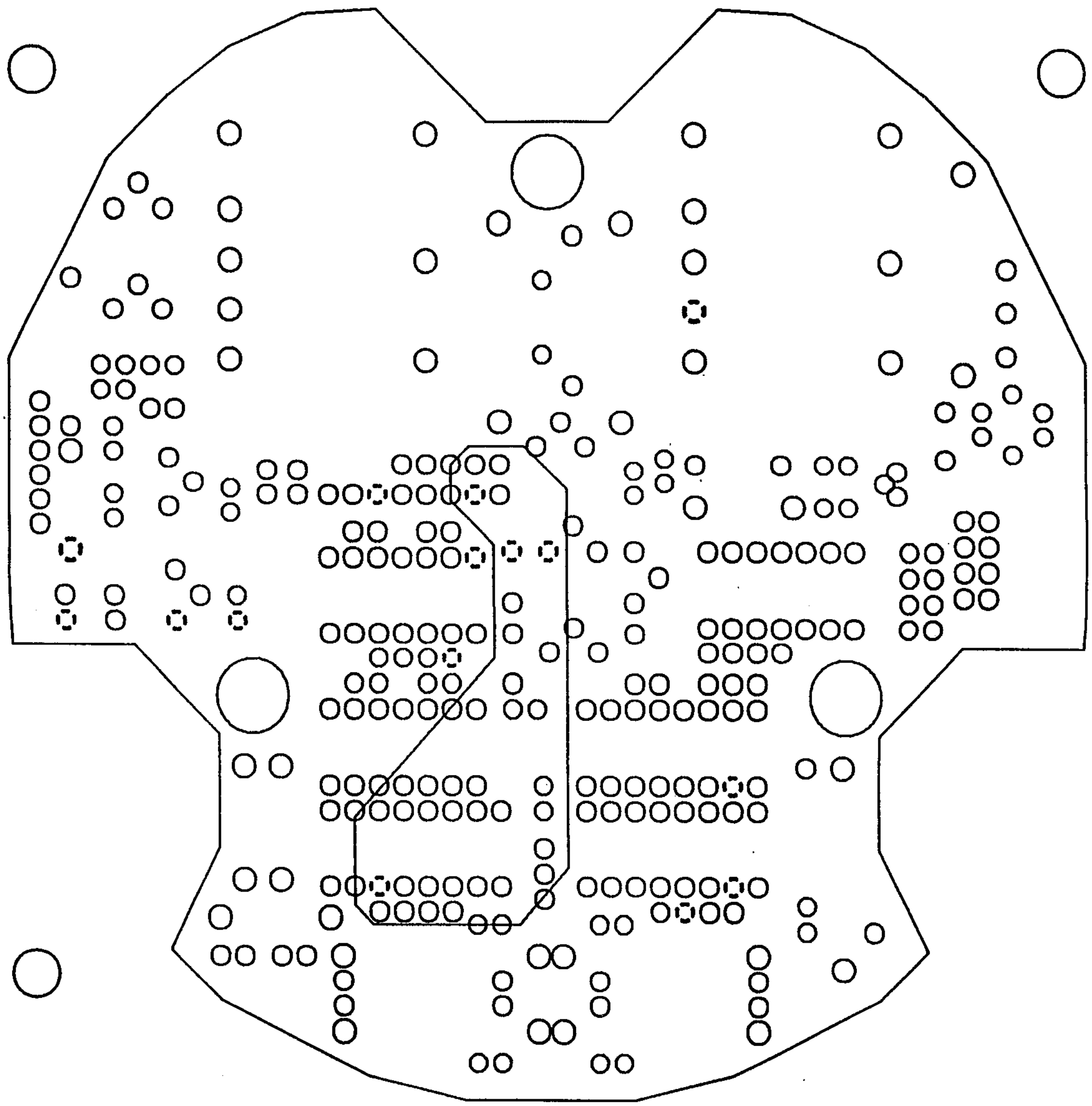
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13014.020		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92

POWER PLANE V-C/VCCA

LAYER 3 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

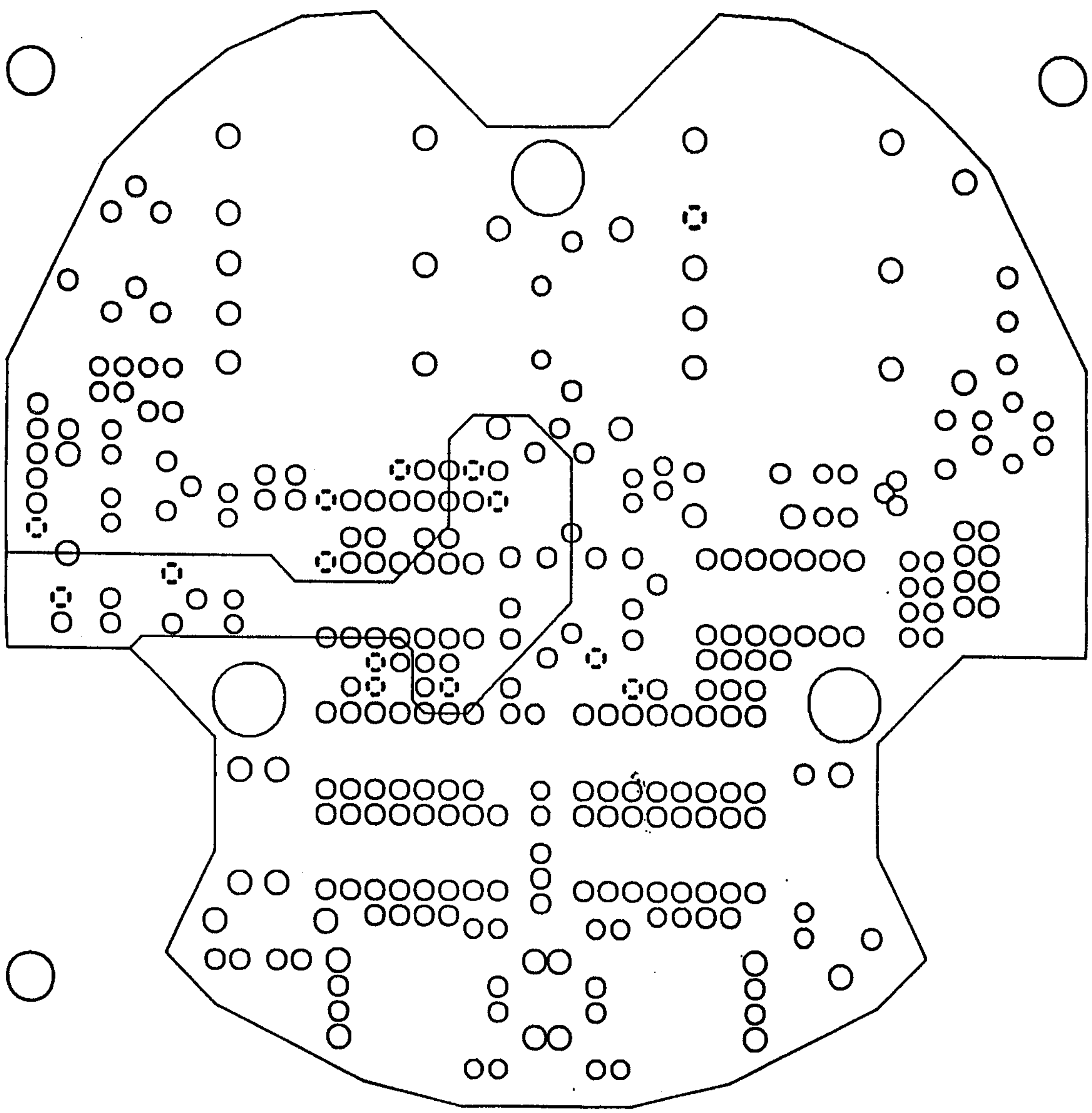
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13014.030		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92

POWER PLANE V-/V+C

LAYER 4 OF 6

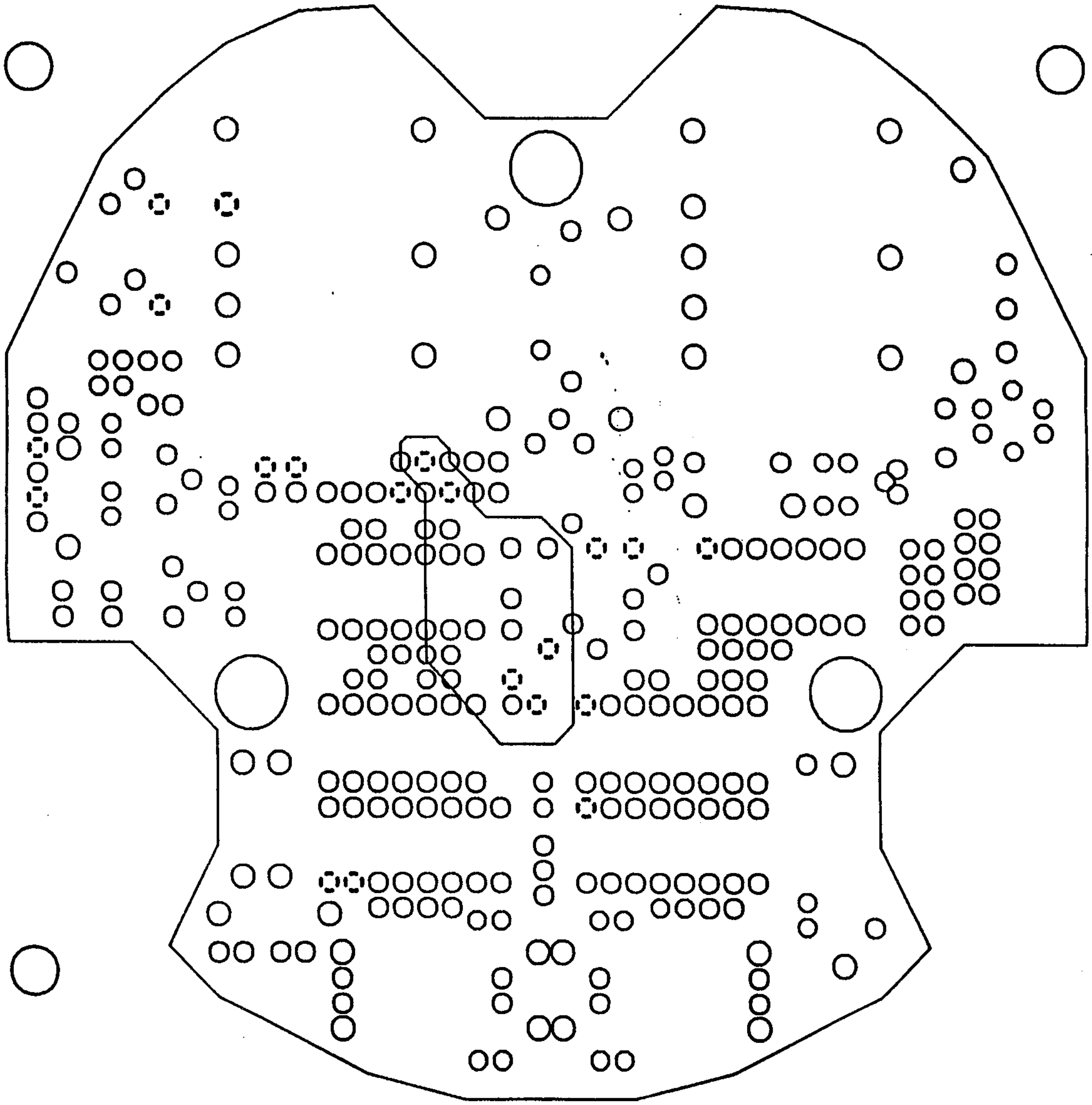


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13014.040		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92

POWER PLANE VCC/V+
LAYER 5 OF 6

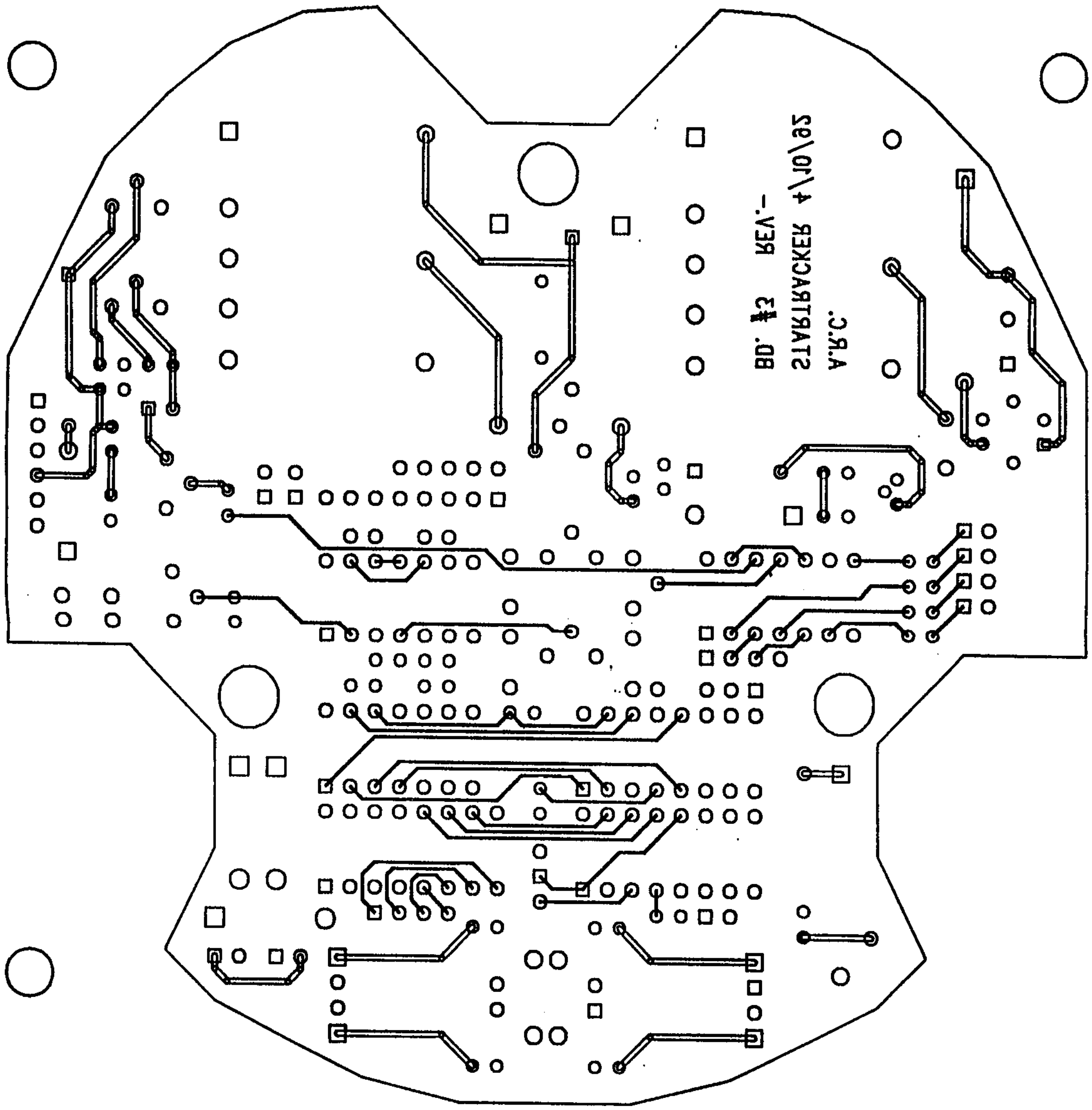


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13014.050		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #3   REV. -   4/10/92

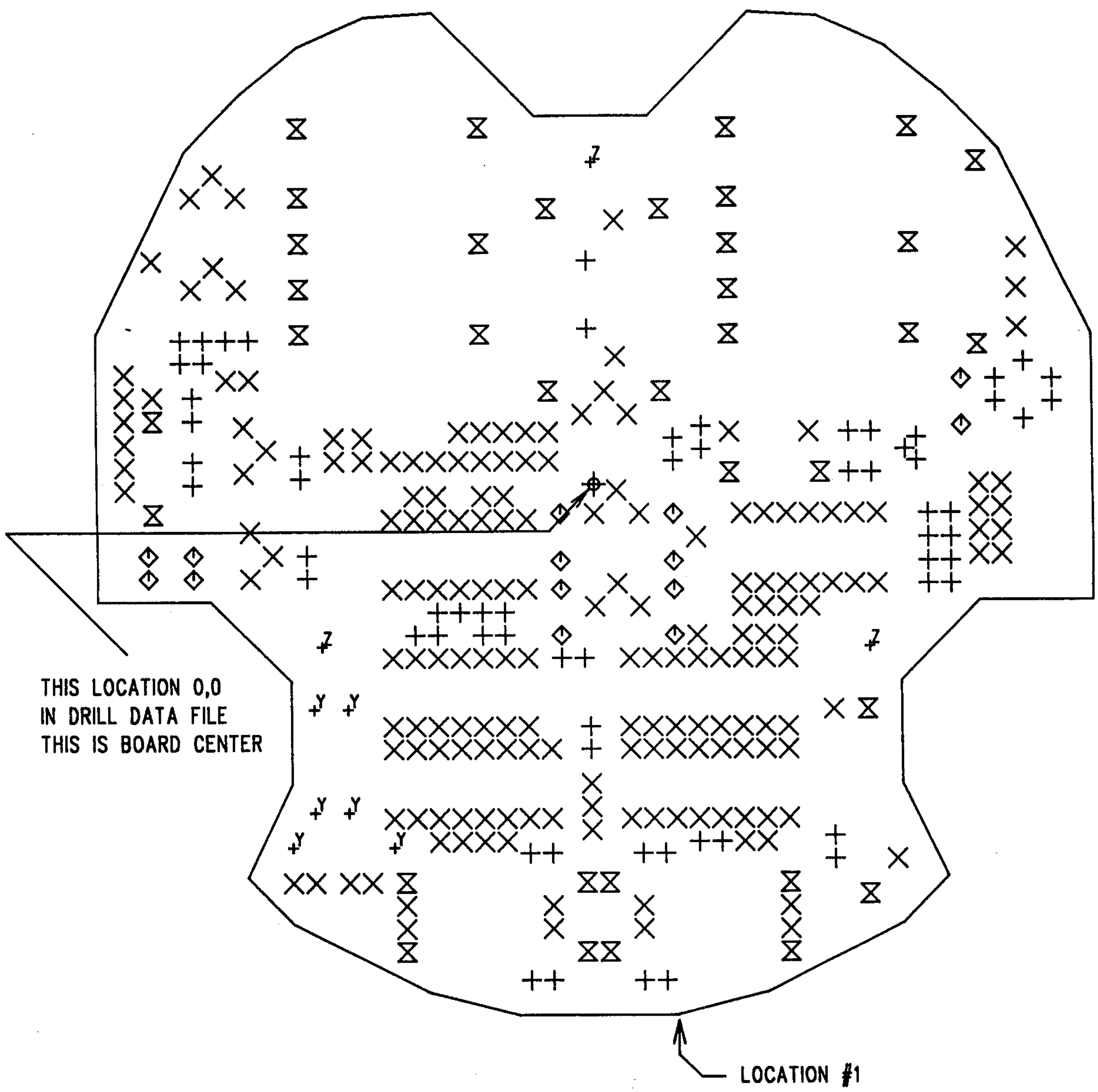
SOLDER SIDE
LAYER 6 OF 6



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13014.060		SHEET NO.

APPLIED RESEARCH CORPORATION DRILL DRAWING

STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92



APPLIED RESEARCH CORP.

8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	
DRAWING NO. D13014.100		SHEET NO.

BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION #1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.900	.600
6.	-1.700	1.000
7.	-1.700	1.450
8.	-2.050	1.800
9.	-2.550	1.800
10.	-2.550	2.950
11.	-2.150	3.750
12.	-1.900	4.000
13.	-1.650	4.200
14.	-1.350	4.330
15.	-1.050	4.350
16.	-.600	3.900
17.	-.100	3.900
18.	.350	4.350
19.	.650	4.330
20.	.950	4.200
21.	1.200	4.000
22.	1.450	3.750
23.	1.850	2.950
24.	1.850	1.800
25.	1.350	1.800
26.	1.000	1.450
27.	1.000	1.000
28.	1.200	.600
29.	1.000	.400
30.	.400	.100
31.	0	0

FABRICATION NOTES:

1. FINISHED BOARD THICKNESS TO BE .062 +/- .007
2. MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
   TYPE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
3. FABRICATE IN ACCORDANCE WITH MIL-P-55110
4. ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
5. LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
   LAYER 1 -COMPONENT SIDE THRU LAYER 6 -SOLDER SIDE
6. MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
7. TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
8. ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
9. ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
   IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
   PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
10. SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
    PER MIL-I-43553

SIZE	QTY	SYM
28	65	+
32	188	X
37	14	◇
42	36	⧖
52	6	Y
200	3	Z

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

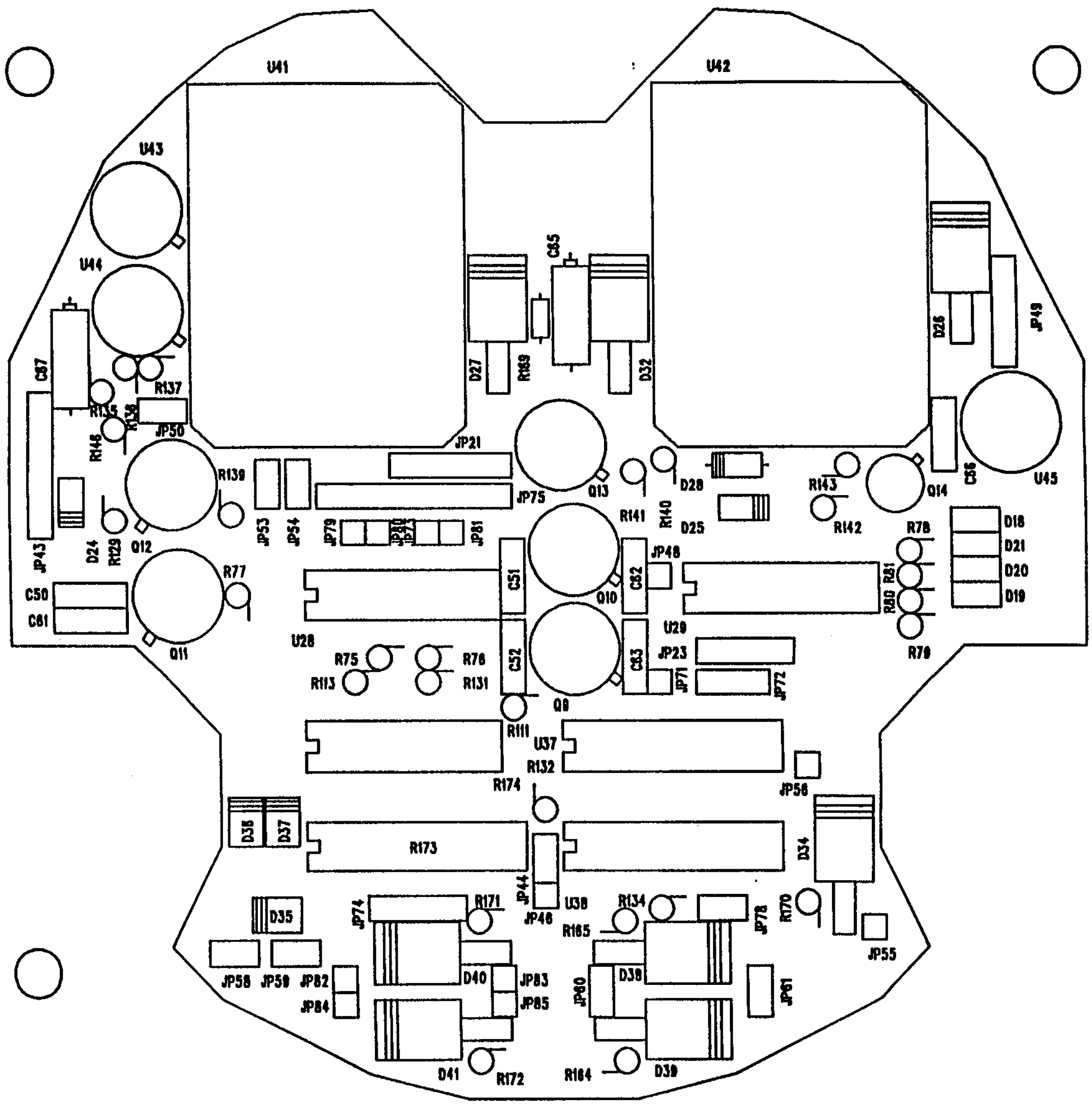
DRAWING NO.	SHEET NO.
D13014.100	

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #3 REV. - 4/10/92

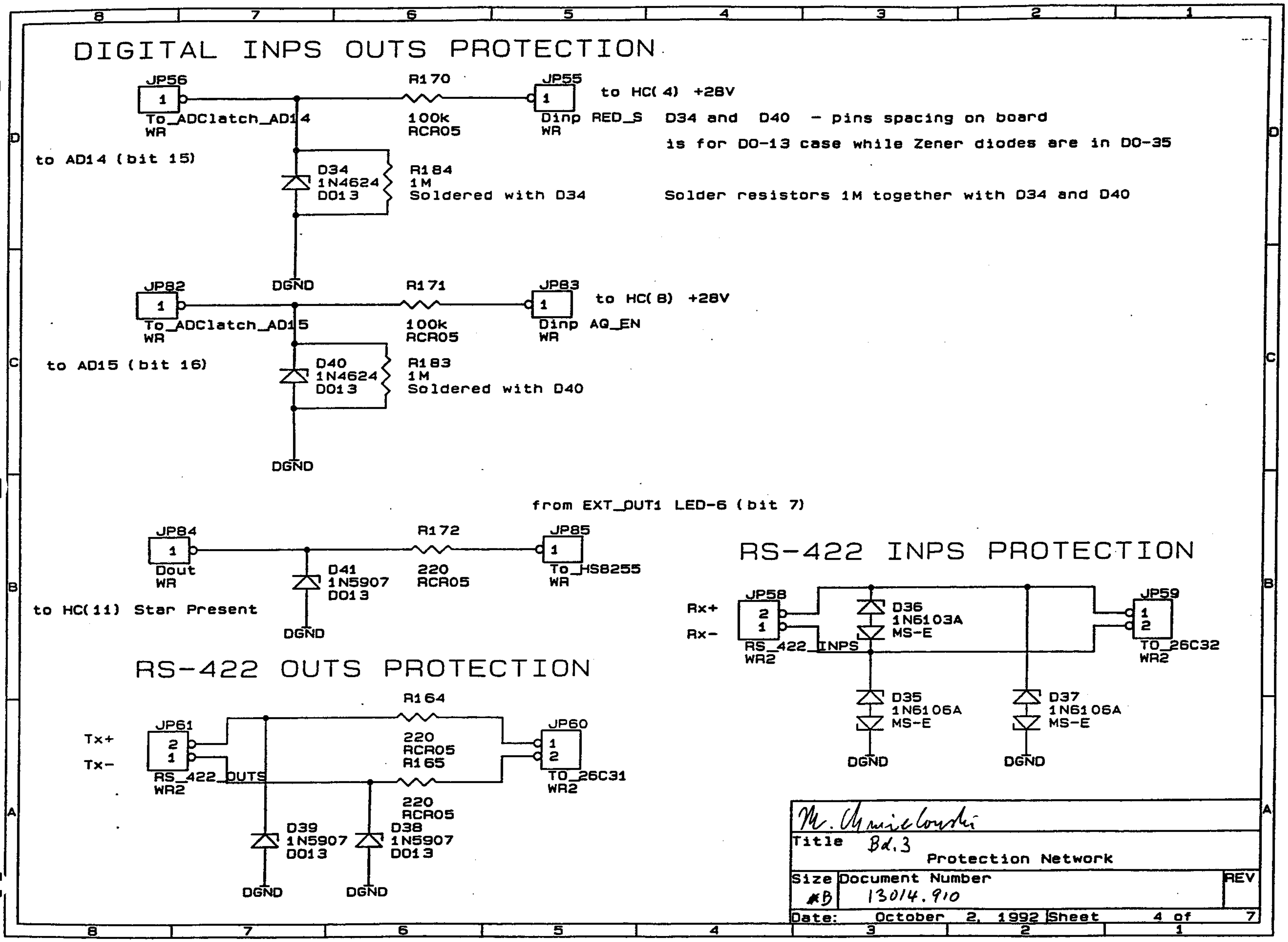
SILKSCREEN

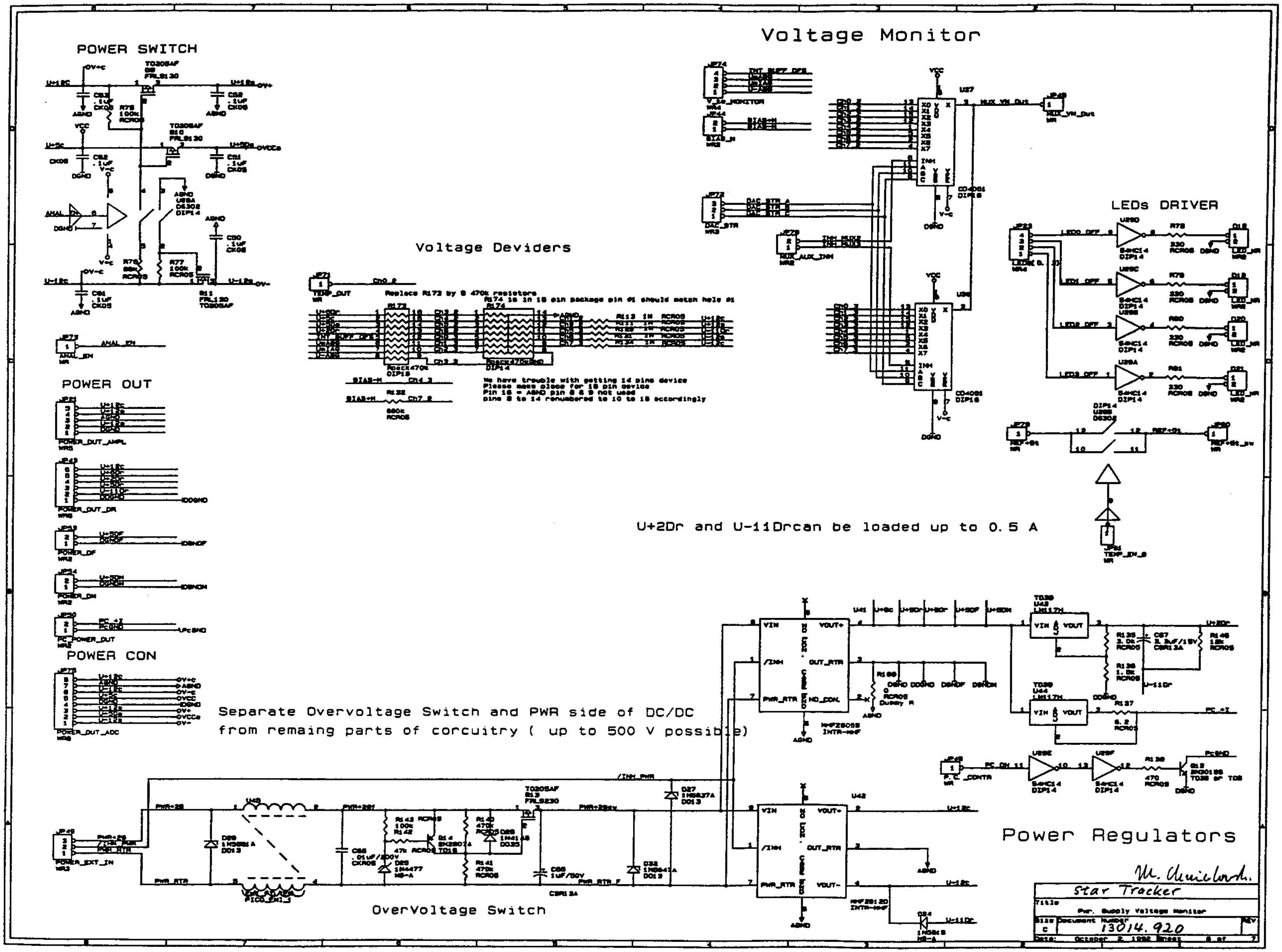
LAYER 1 OF 4



Use or disclosure of these
SBIR data is subject to the
restriction on the title page
of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13014.110		SHEET NO.


DIGITAL INPS OUTS PROTECTION
JP56
To_ADClatch_AD14
WR
to AD14 (bit 15)
R170
100k
RCR05
JP55
to HC(4) +28V
Dinp RED_S
WR
D34 and D40 - pins spacing on board
is for DO-13 case while Zener diodes are in DO-35
D34
1N4624
DO13
R184
1M
Soldered with D34
Solder resistors 1M together with D34 and D40
DGND
JP82
To_ADClatch_AD15
WR
to AD15 (bit 16)
R171
100k
RCR05
JP83
to HC(8) +28V
Dinp AQ_EN
WR
D40
1N4624
DO13
R183
1M
Soldered with D40
DGND
from EXT_OUT1 LED-6 (bit 7)
JP84
Dout
WR
to HC(11) Star Present
D41
1N5907
DO13
DGND
R172
220
RCR05
JP85
To_HS8255
WR
RS-422 INPS PROTECTION
Rx+
Rx-
JP58
RS_422_INPS
WR2
D36
1N6103A
MS-E
JP59
TO_26C32
WR2
D35
1N6106A
MS-E
D37
1N6106A
MS-E
DGND
RS-422 OUTS PROTECTION
Tx+
Tx-
JP61
RS_422_OUTS
WR2
R164
220
RCR05
R165
220
RCR05
JP60
TO_26C31
WR2
D39
1N5907
DO13
D38
1N5907
DO13
DGND
Title Bd.3
Protection Network
Size B
Document Number 13014.910
REV
Date: October 2, 1992 Sheet 4 of 7

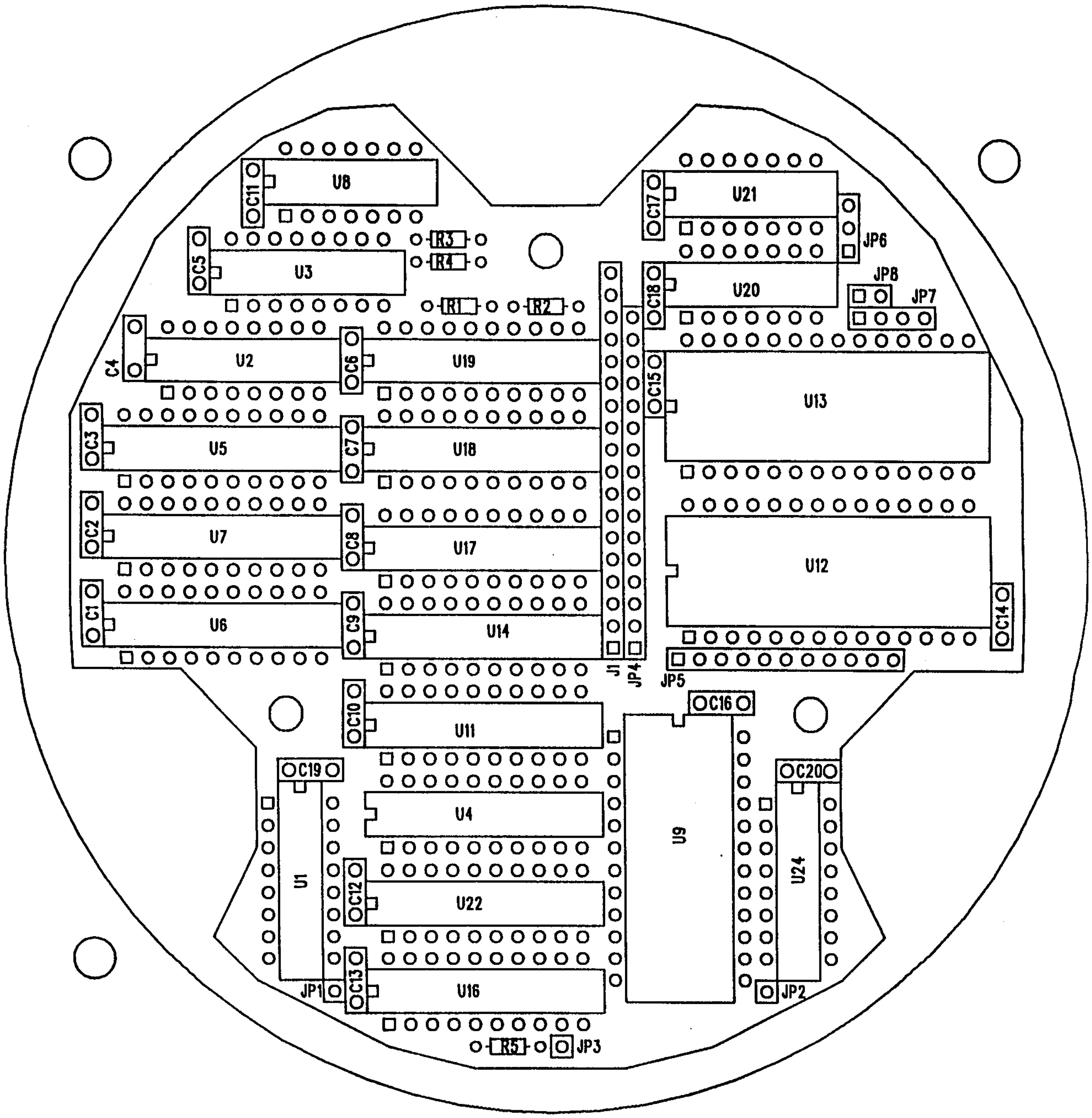


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORPORATION

STARTRACKER PCSG BOARD(#4) REV. - 7/28/92

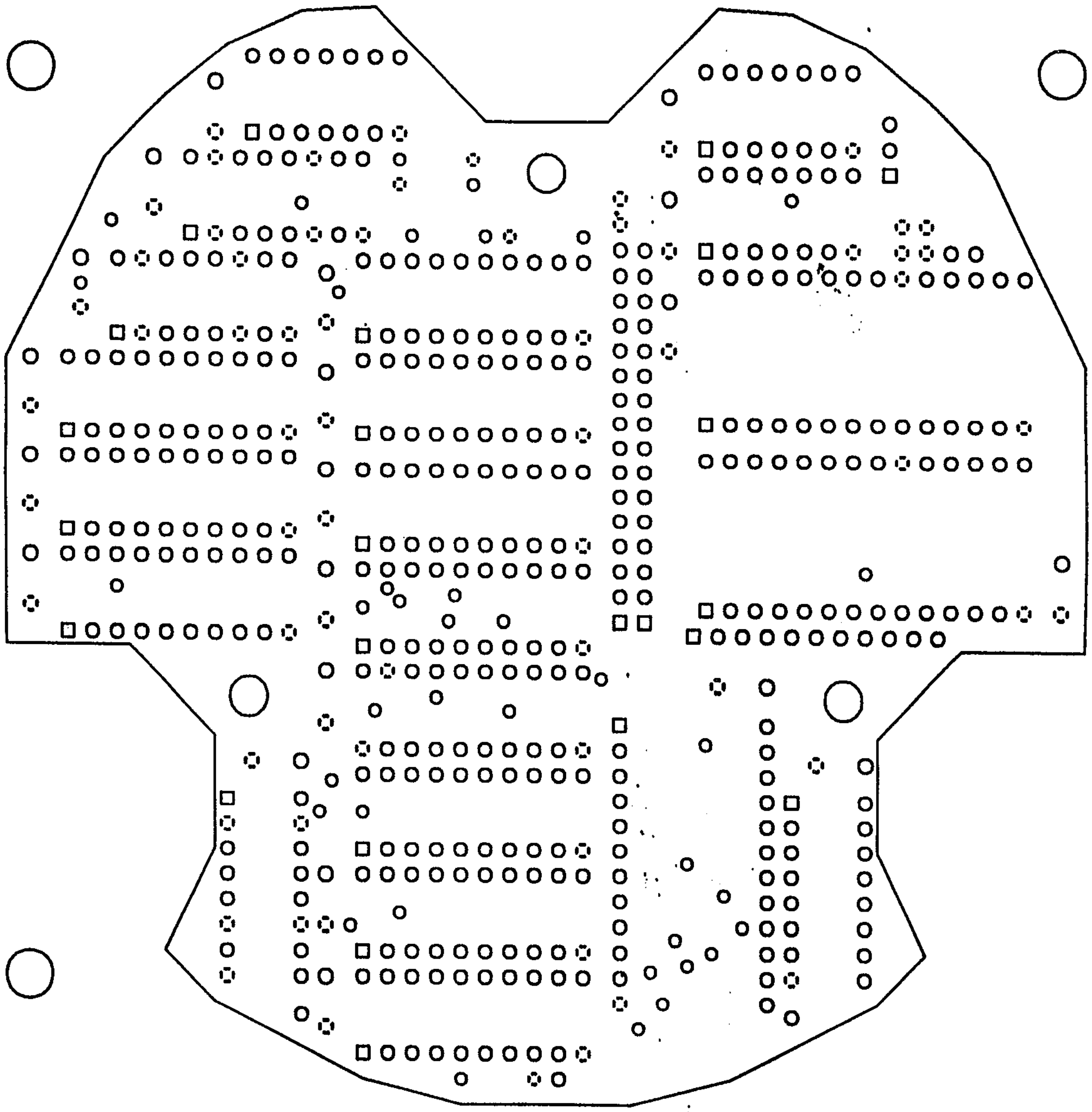
ASSEMBLY DRAWING



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.000		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD(#4) REV. - 7/28/92
COMP. SIDE
GROUND PLANE
LAYER 1 OF 6

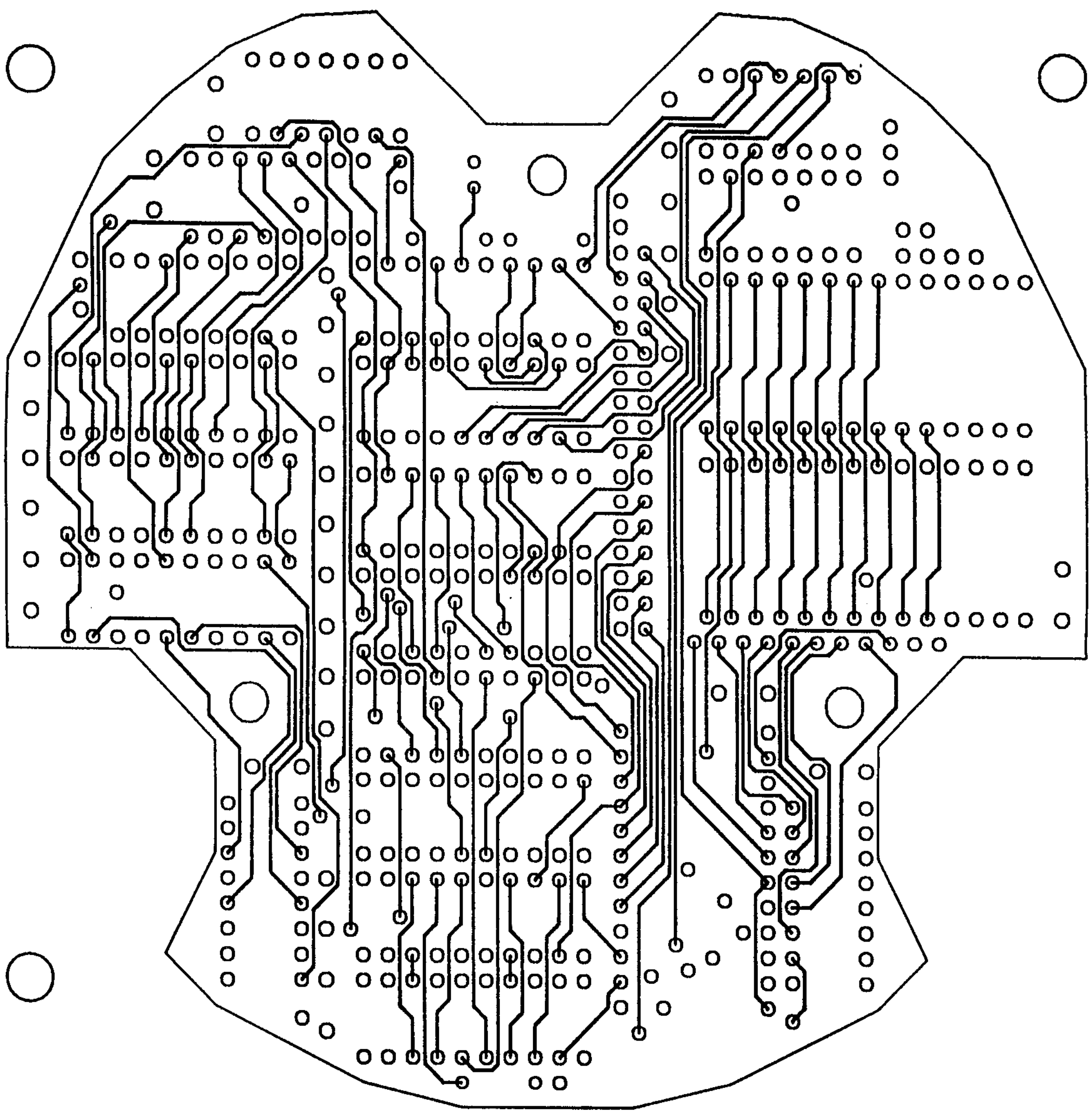


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.010		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD(#4) REV. - 7/28/92

SIGNAL LAYER
LAYER 2 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

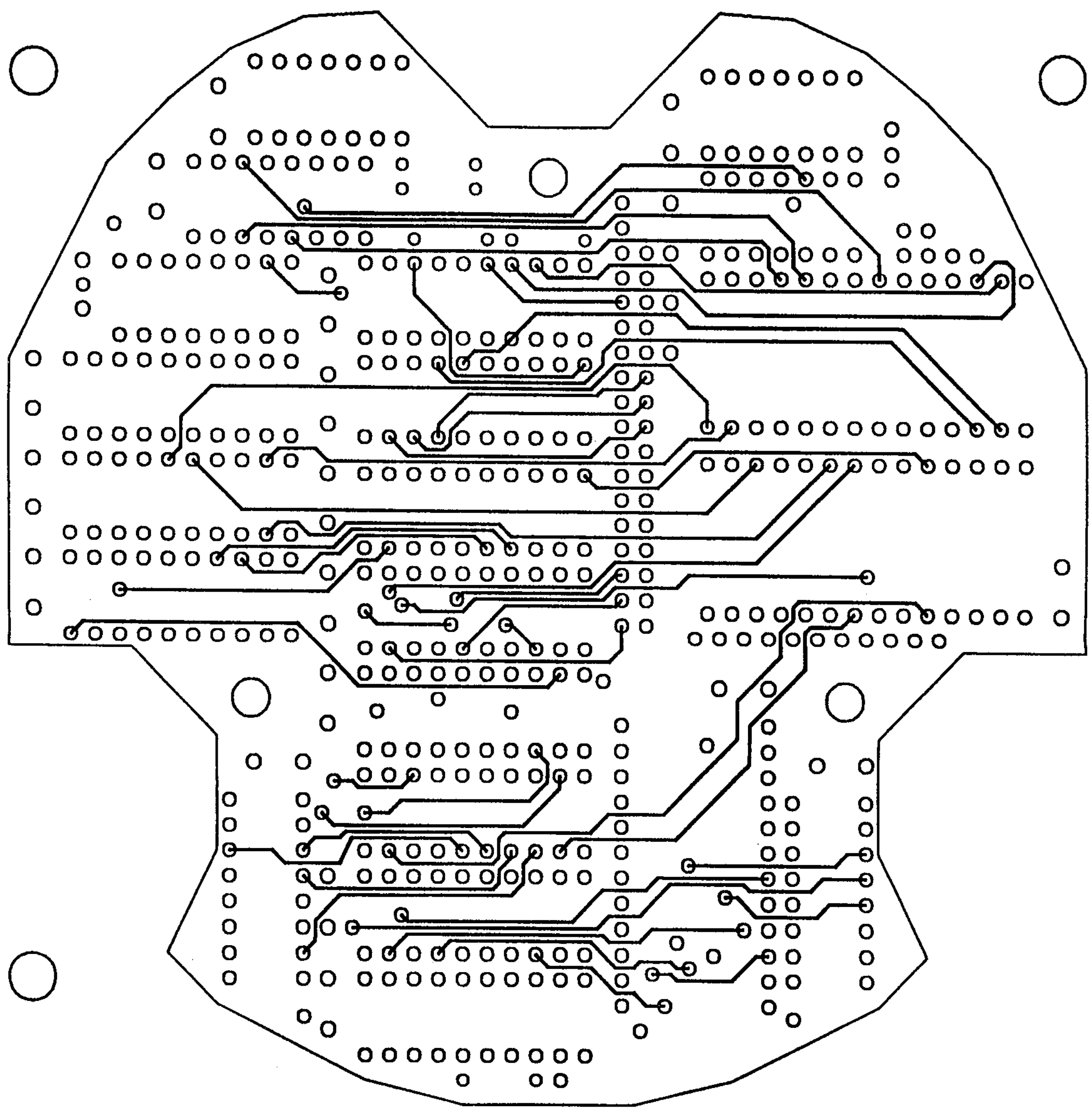
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.020		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER PCSG BOARD(#4)REV. - 7/28/92

SIGNAL LAYER

LAYER 3 OF 6

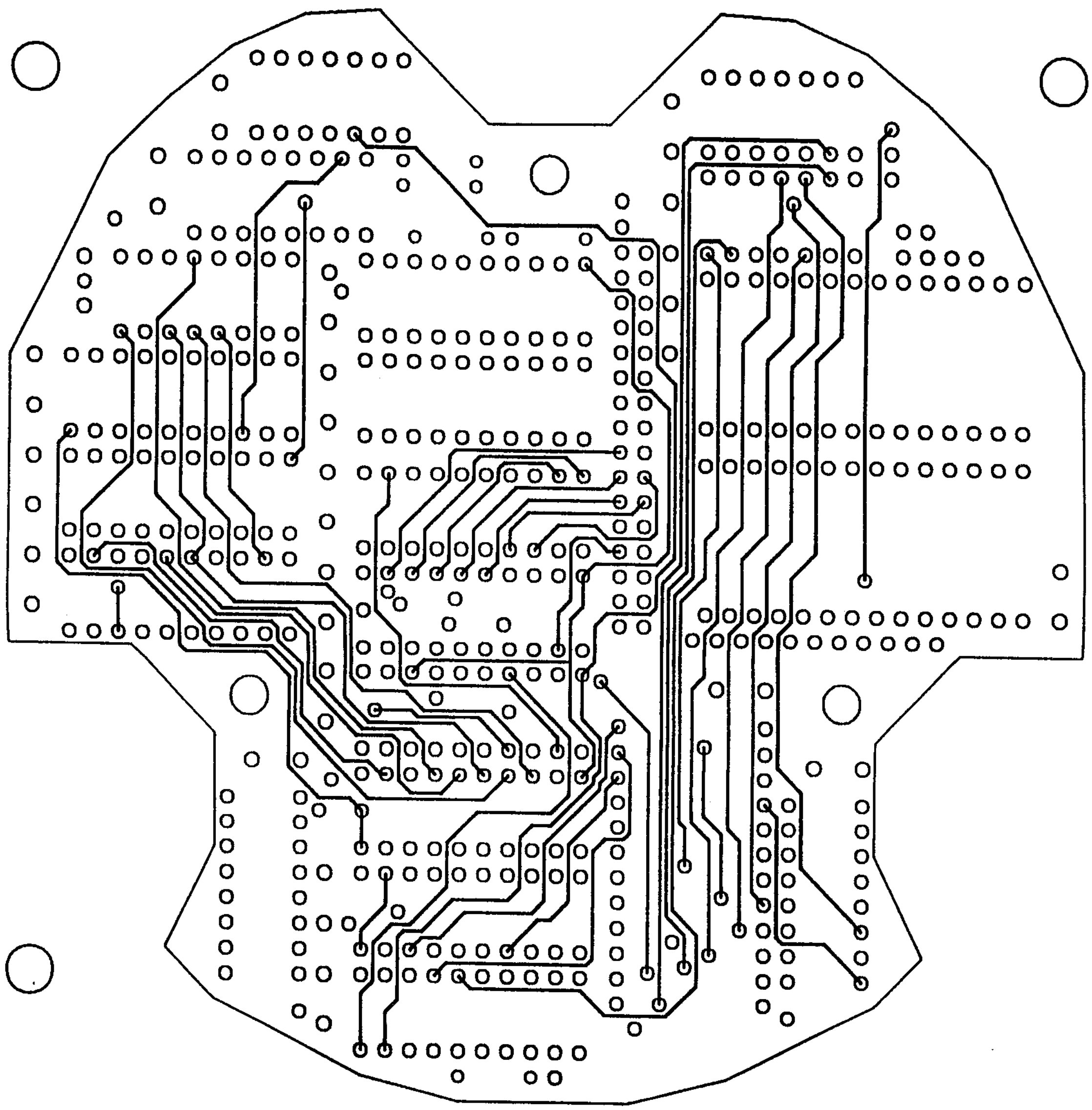


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.030		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD(#4)REV. - 7/28/92

SIGNAL LAYER
LAYER 4 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

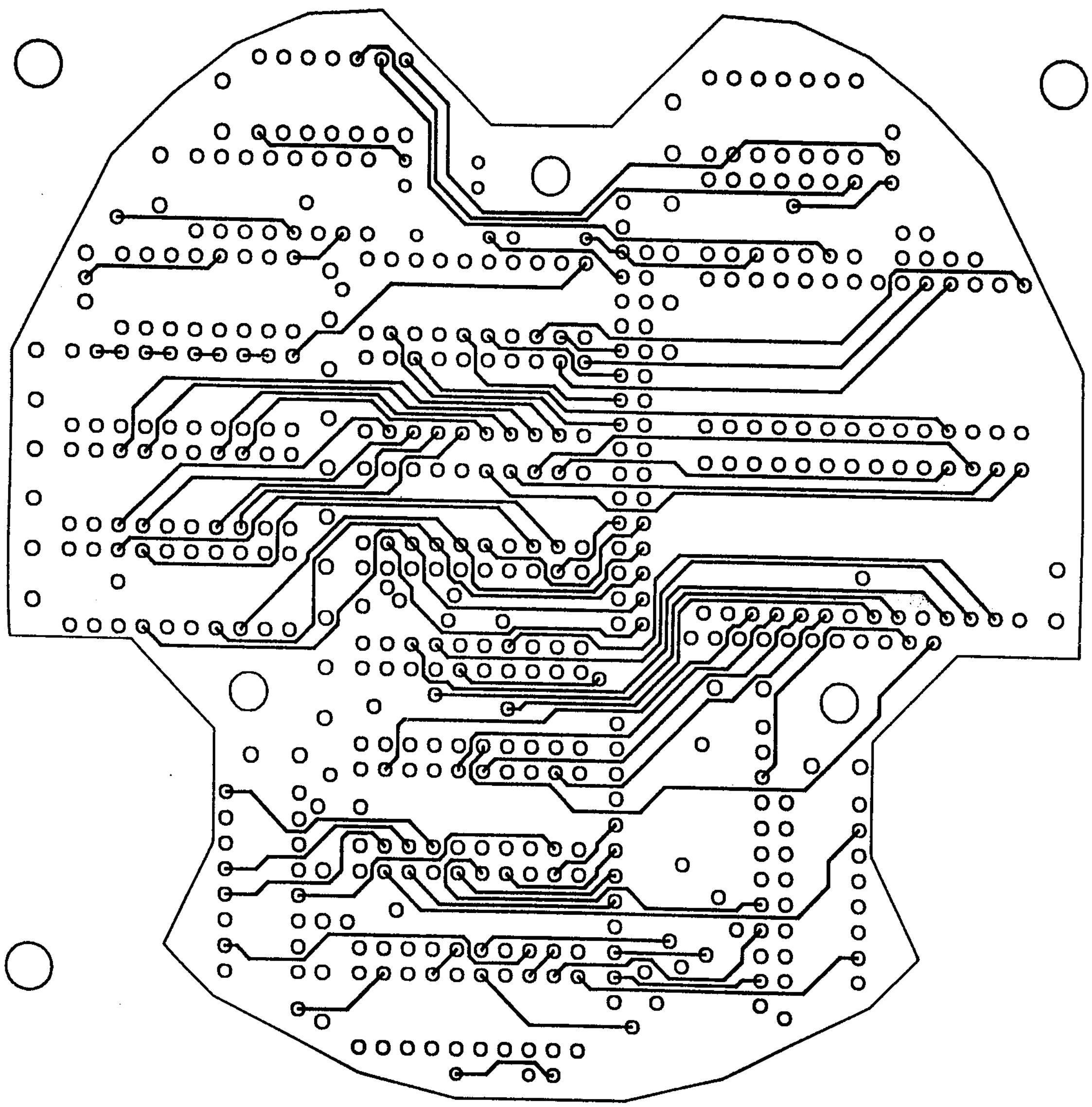
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.040		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER PCSG BOARD(#4)REV. - 7/28/92

SIGNAL LAYER

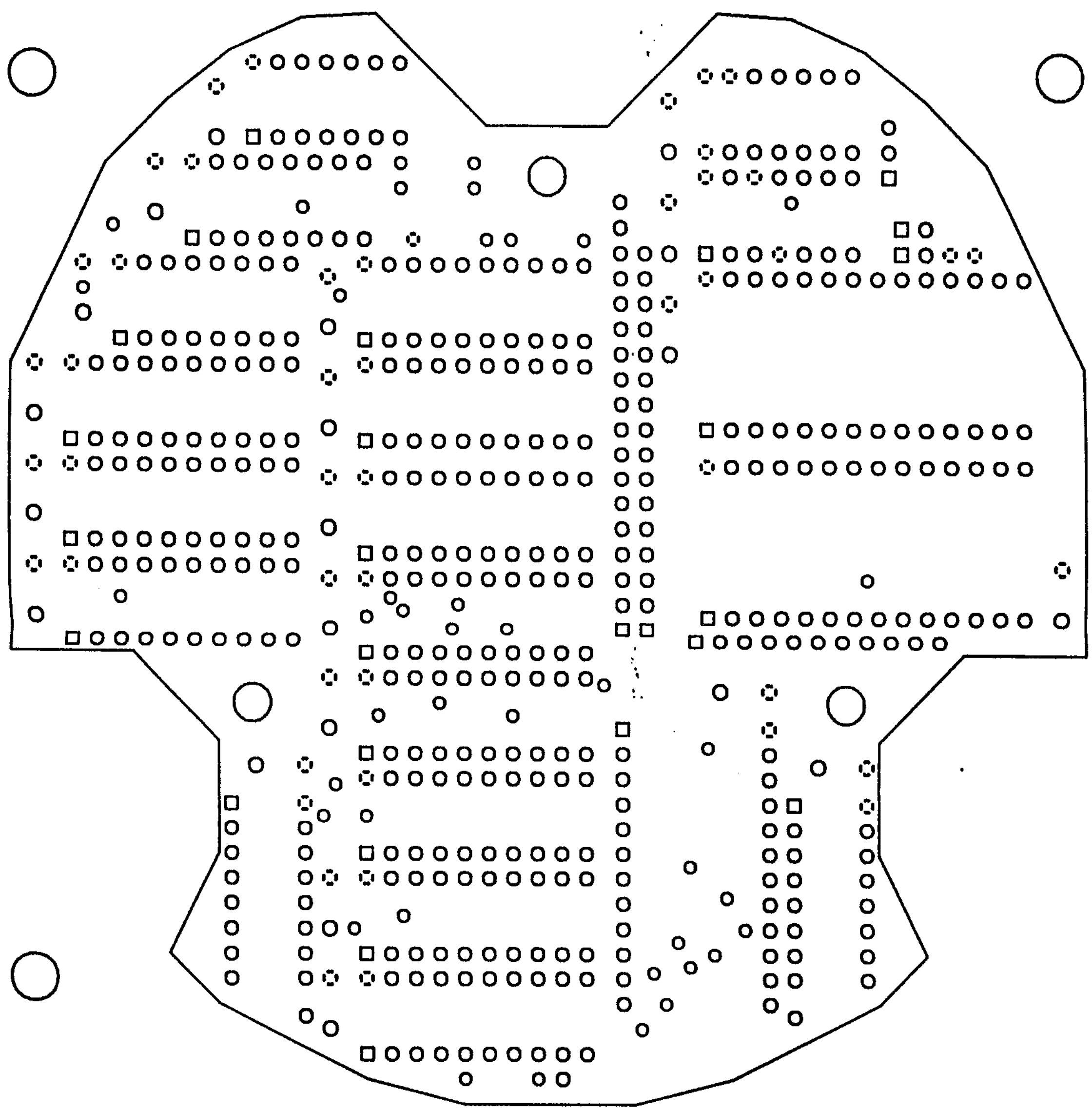
LAYER ~~4~~ 5 OF ~~4~~ 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13015.050		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD(#4)REV. - 7/28/92
SOLDER SIDE
POWER PLANE
LAYER 6 OF 6

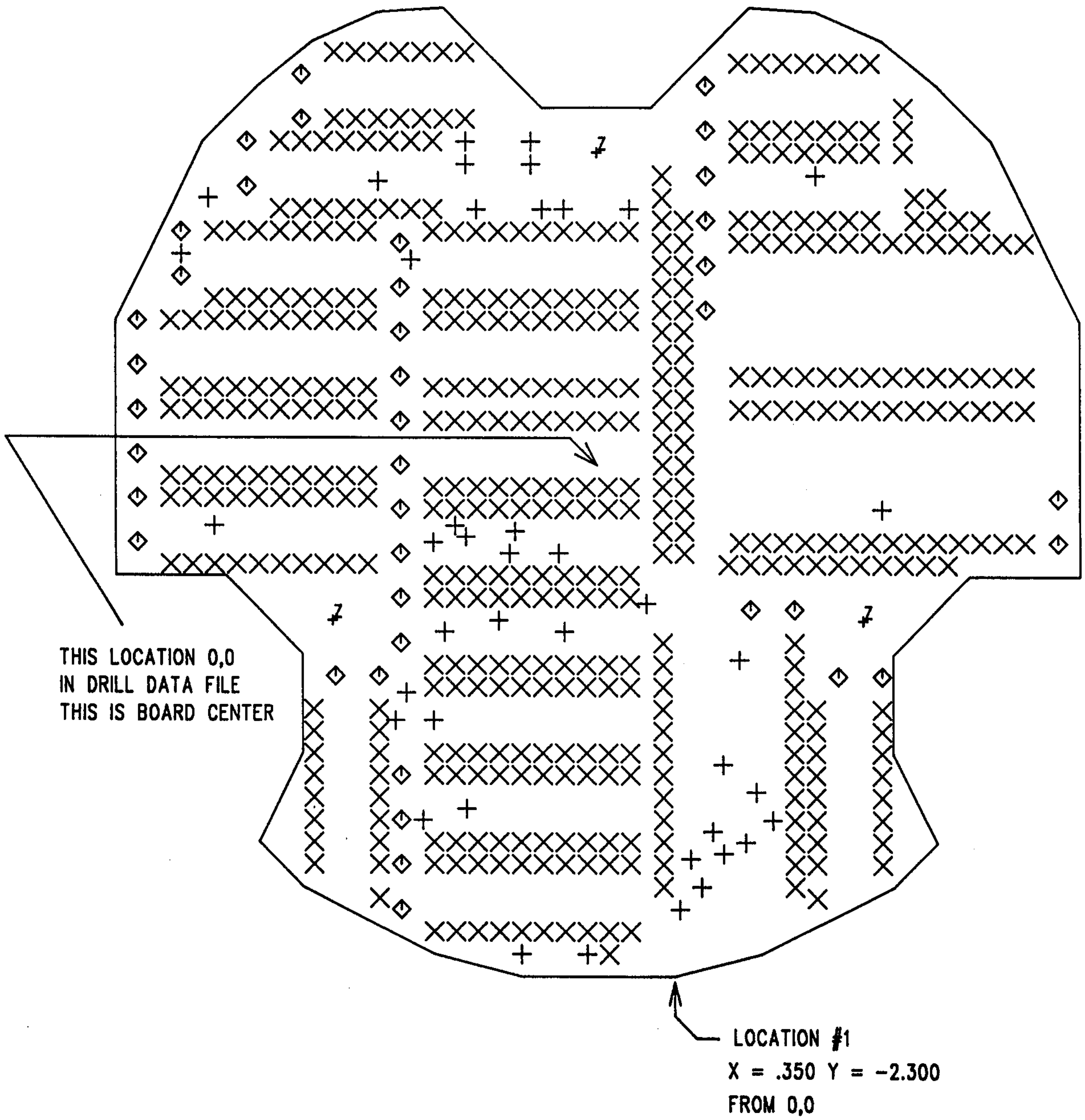


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	
DRAWING NO.	B13015.060	SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD REV. - 7/28/92
PCB #4
DRILL DRAWING



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. D 13015.100		SHEET NO.

SIZE	QTY	SYM
28	42	+
32	483	X
37	40	◇
160	3	Z

BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION #1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.900	.600
6.	-1.700	1.000
7.	-1.700	1.450
8.	-2.050	1.800
9.	-2.550	1.800
10.	-2.550	2.950
11.	-2.150	3.750
12.	-1.900	4.000
13.	-1.650	4.200
14.	-1.350	4.330
15.	-1.050	4.350
16.	-.600	3.900
17.	-.100	3.900
18.	.350	4.350
19.	.650	4.330
20.	.950	4.200
21.	1.200	4.000
22.	1.450	3.750
23.	1.850	2.950
24.	1.850	1.800
25.	1.350	1.800
26.	1.000	1.450
27.	1.000	1.000
28.	1.200	.600
29.	1.000	.400
30.	.400	.100
31.	0	0

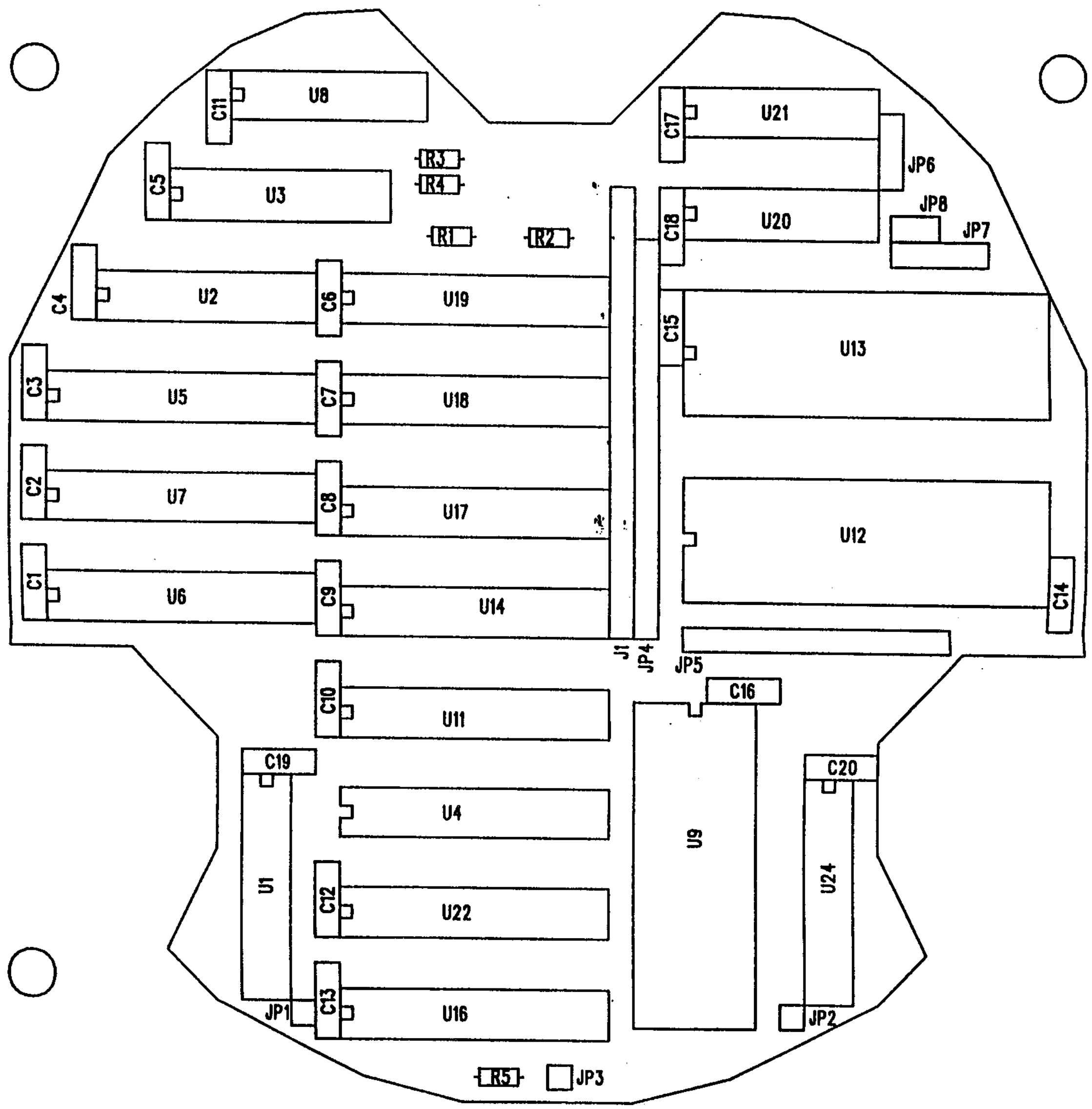
FABRICATION NOTES:

1. FINISHED BOARD THICKNESS TO BE .062 +/- .007
2. MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
   TYRE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
3. FABRICATE IN ACCORDANCE WITH MIL-P-55110
4. ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
5. LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
   LAYER 1 -COMPONENT SIDE THRU LAYER 6 -SOLDER SIDE
6. MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
7. TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
8. ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
9. ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
   IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
   PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
10. ALL .160" DIA. HOLES ARE TO BE NON PLATED THRU 3 PLCS.
11. SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
    PER MIL-I-43553

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. D 13015.100		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER PCSG BOARD(#4)REV. - 7/28/92

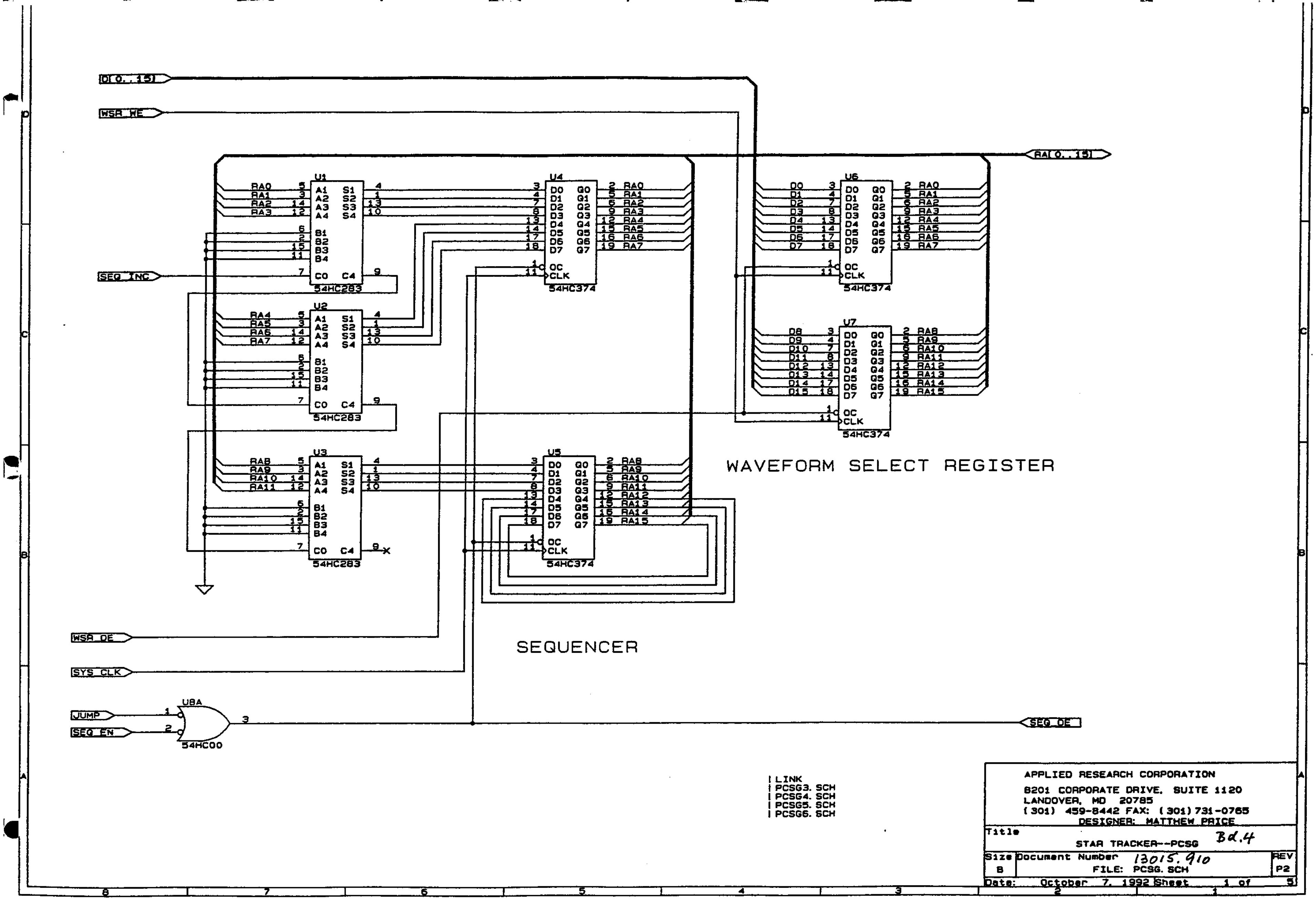
SILKSCREEN
LAYER 1 OF 6



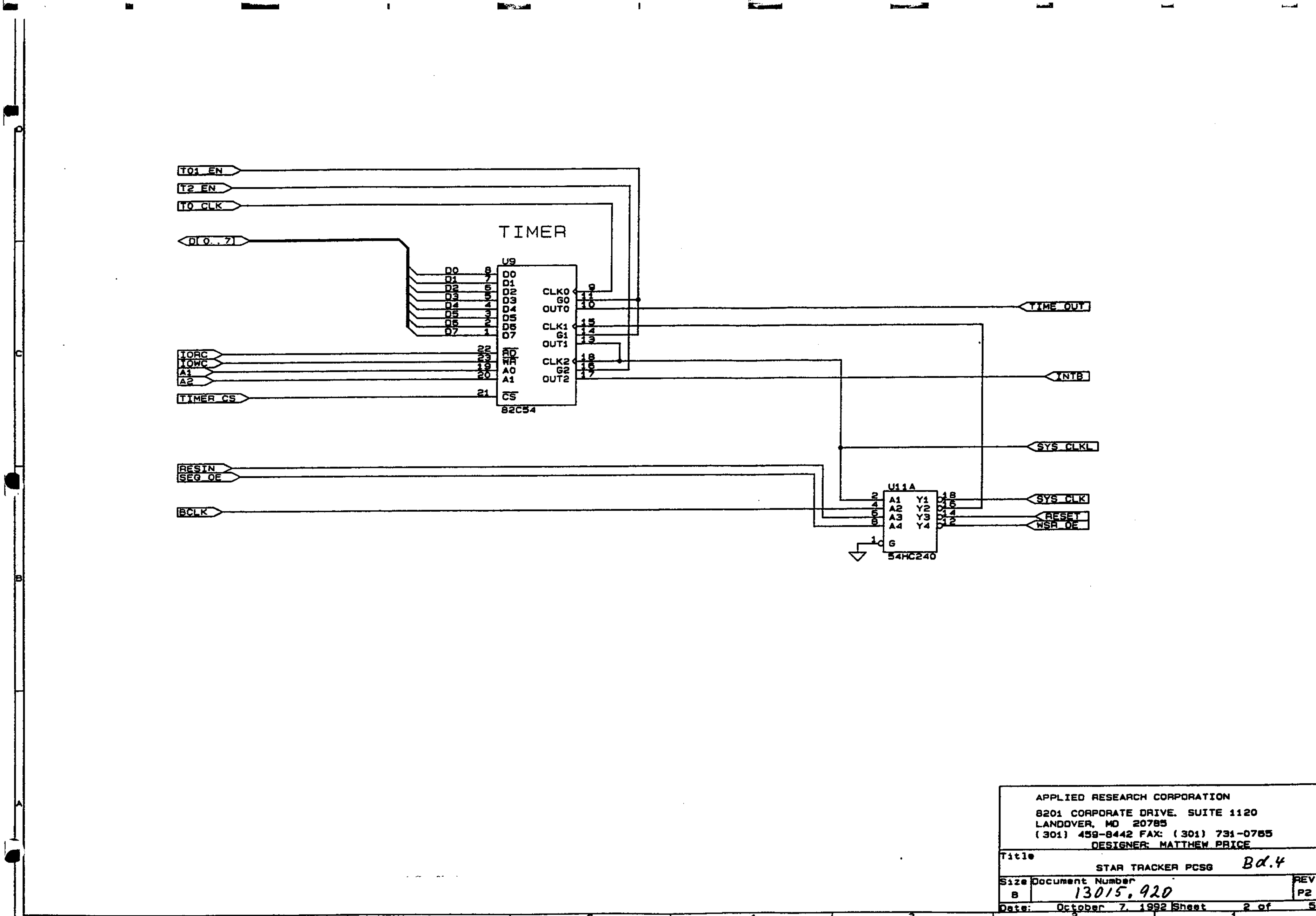
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
B 13015.110	



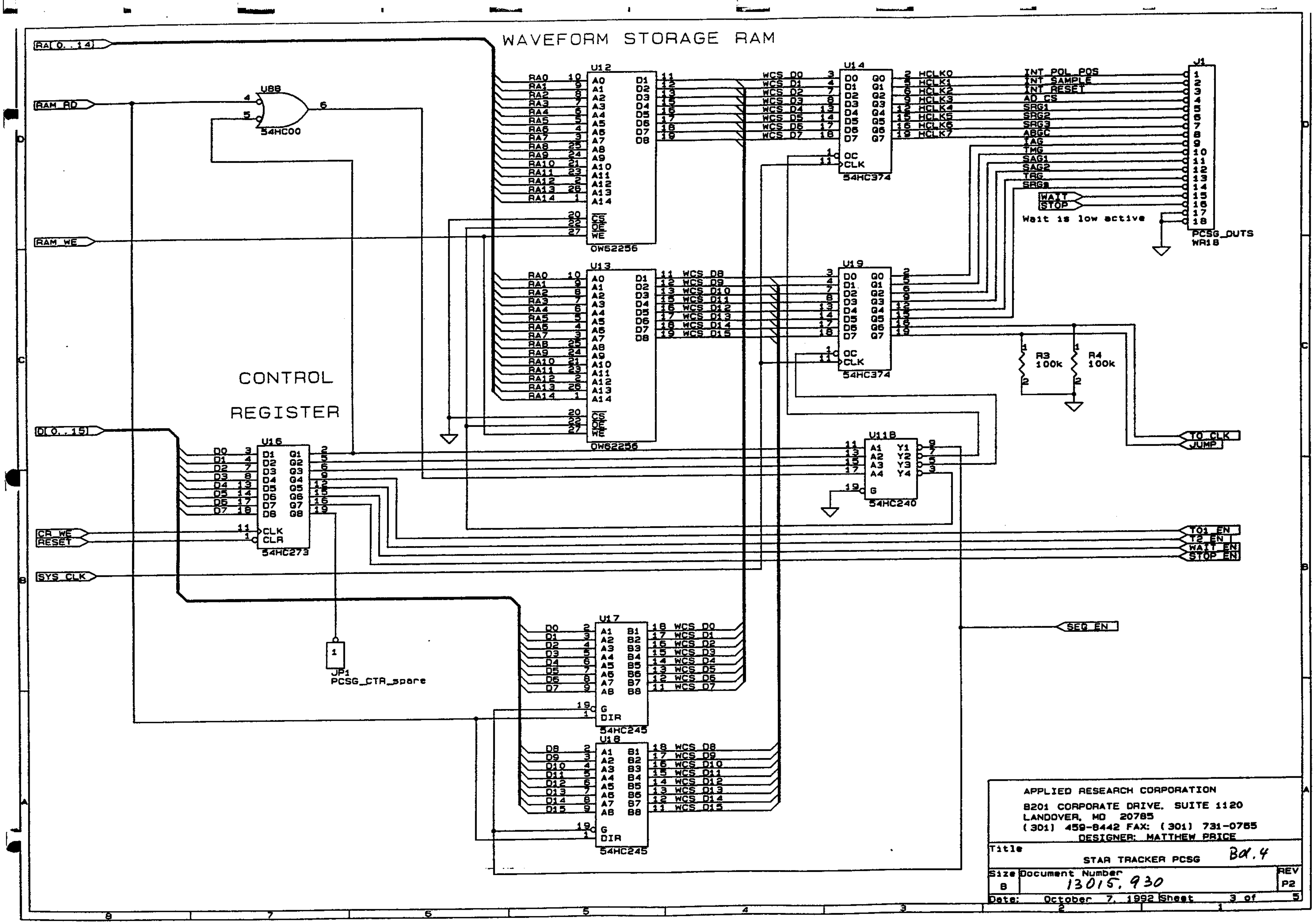
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



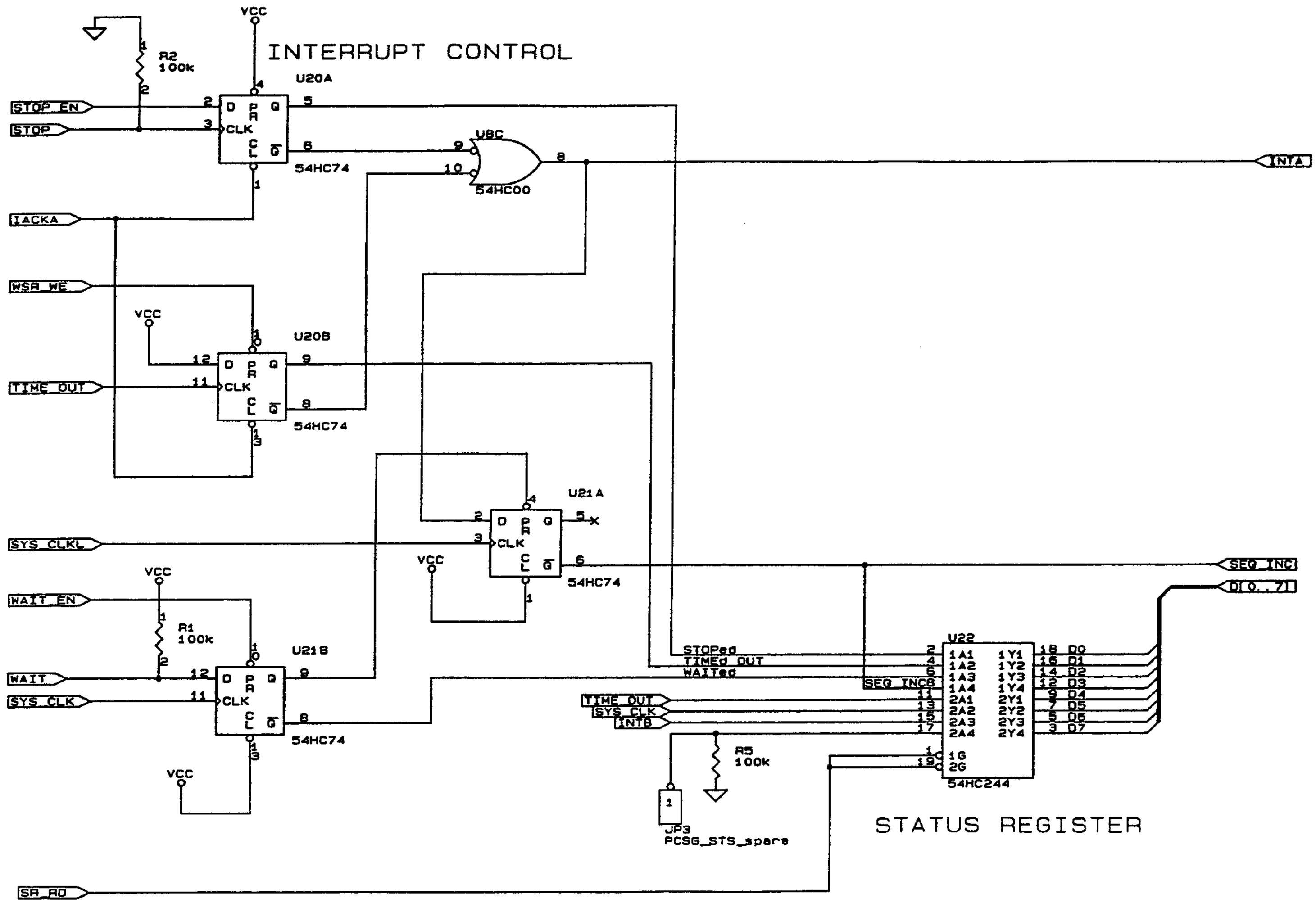
APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE, SUITE 1120
LANDOVER, MD 20785
(301) 459-8442 FAX: (301) 731-0765
DESIGNER: MATTHEW PRICE

Title: STAR TRACKER PCSG Bd.4		
Size: B	Document Number: 13015.920	REV: P2
Date: October 7, 1992	Sheet 2 of 5	

Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



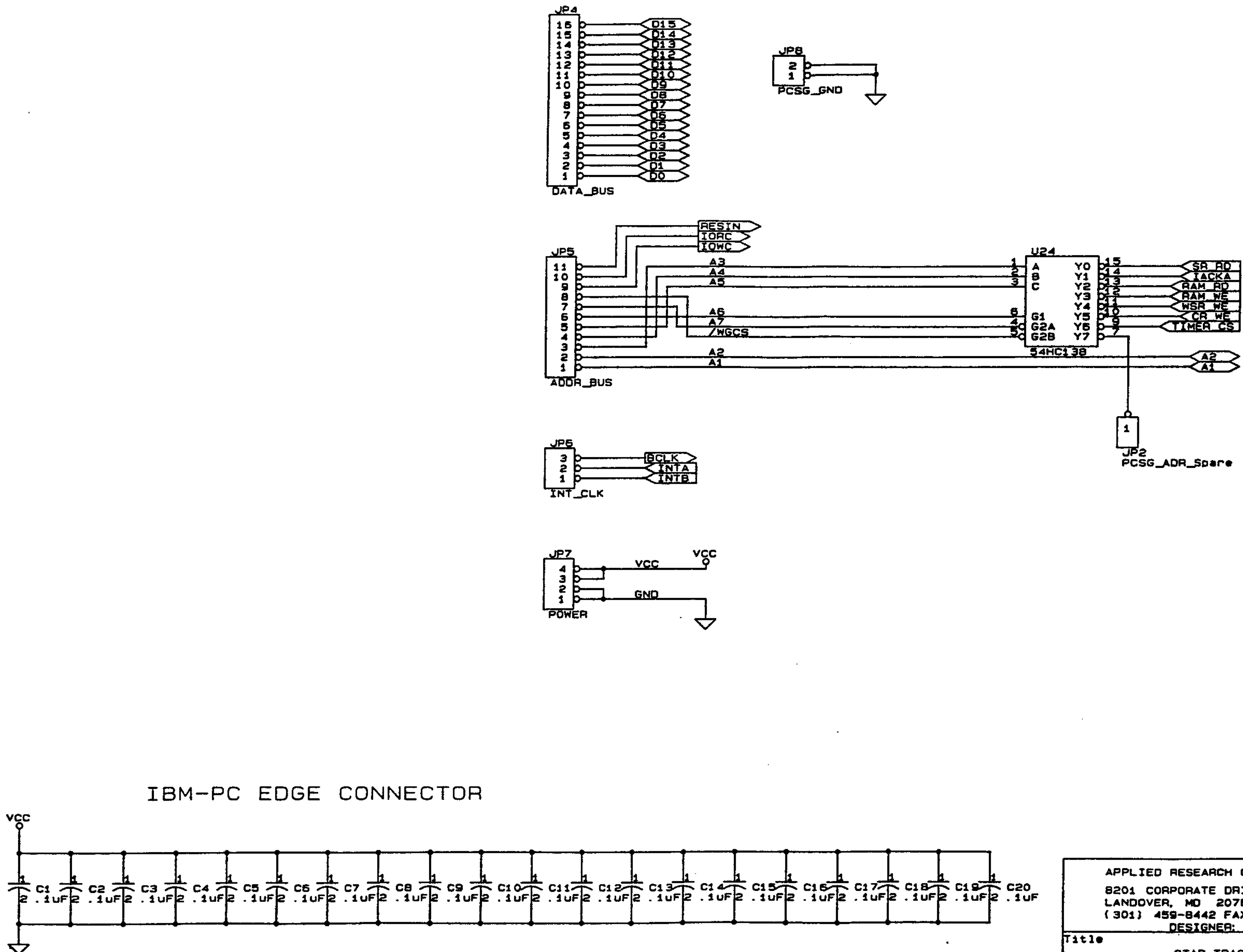
WAVEFORM STORAGE RAM
CONTROL REGISTER
RA[0..14]
RAM RD
RAM WE
D[0..15]
CR WE
RESET
SYS CLK
U8B 54HC00
U12 OW62256
U13 OW62256
U14 54HC374
U19 54HC374
U16 54HC273
U11B 54HC240
U17 54HC245
U18 54HC245
J1 PCSG_OUTS WR18
INT POL POS
INT SAMPLE
INT RESET
AD CS
SRG1
SRG2
SRG3
ABGC
IAG
TMG
SAG1
SAG2
TRG
SRGs
WAIT
STOP
Wait is low active
R3 100k
R4 100k
TO CLK
JUMP
TO1 EN
T2 EN
WAIT EN
STOP EN
SEG EN
JP1 PCSG_CTR_spare
APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE, SUITE 1120
LANDOVER, MD 20785
(301) 459-8442 FAX: (301) 731-0765
DESIGNER: MATTHEW PRICE
Title
STAR TRACKER PCSG
Bd. 4
Size B
Document Number 13015.930
REV P2
Date: October 7, 1992 Sheet 3 of 5



APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE, SUITE 1120
LANDOVER, MD 20785
(301) 459-8442 FAX: (301) 731-0765
DESIGNER: MATTHEW PRICE

Title	STAR TRACKER PCSG Bd. 4	
Size B	Document Number 13015.940	REV P2
Date: October 7, 1992	Sheet 4 of 5	

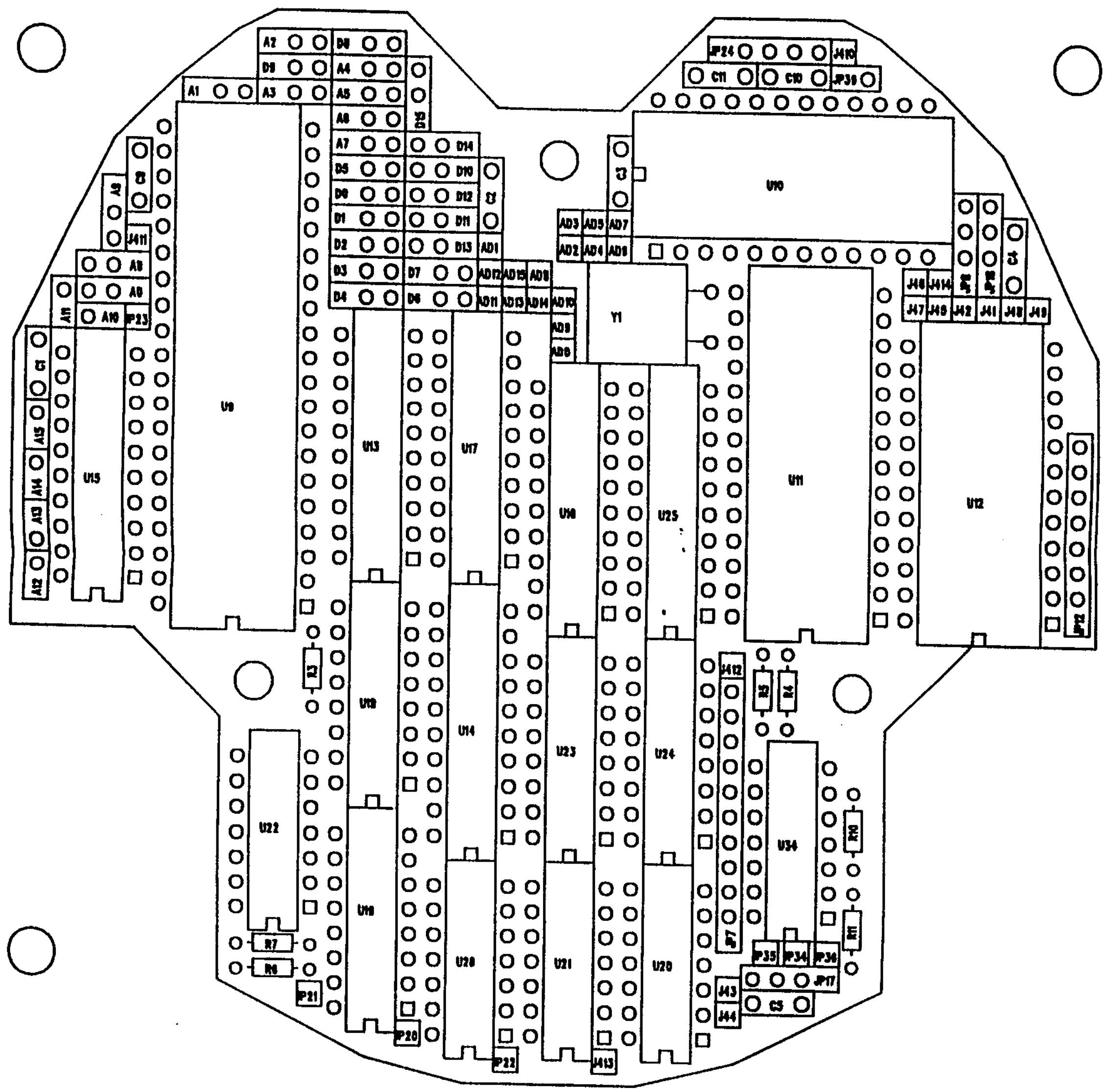
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.



APPLIED RESEARCH CORPORATION
8201 CORPORATE DRIVE, SUITE 1120
LANDOVER, MD 20785
(301) 459-8442 FAX: (301) 731-0765
DESIGNER: MATTHEW PRICE

Title	STAR TRACKER PCSG Bd. 4	
Size B	Document Number 13015.950	REV P2
Date: October 7, 1992	Sheet 5 of 5	

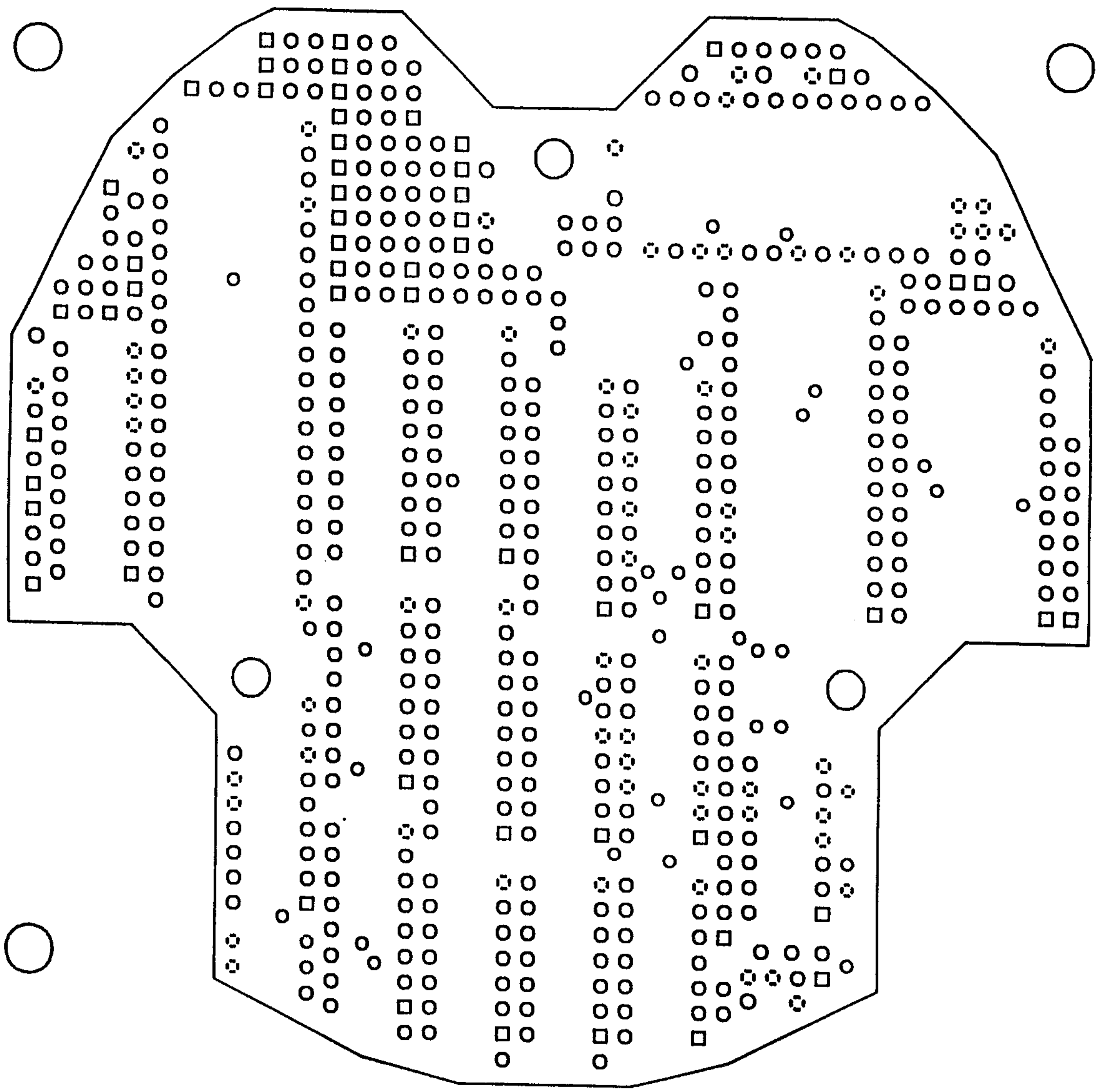
APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92
ASSEMBLY DRAWING



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
B1306.000		

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92
GROUND PLANE
LAYER 1 OF 6



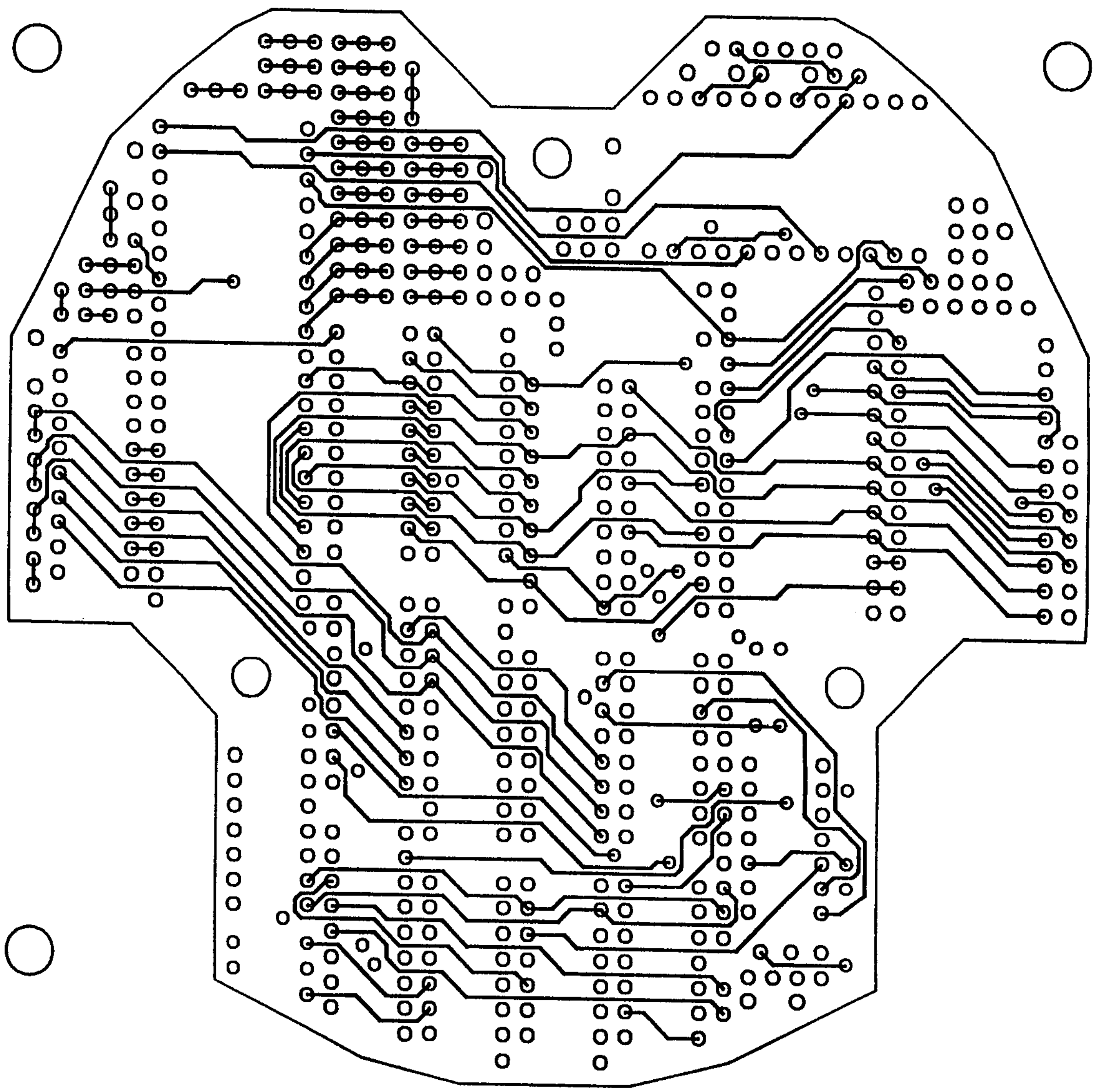
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.010		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92

SIGNAL LAYER

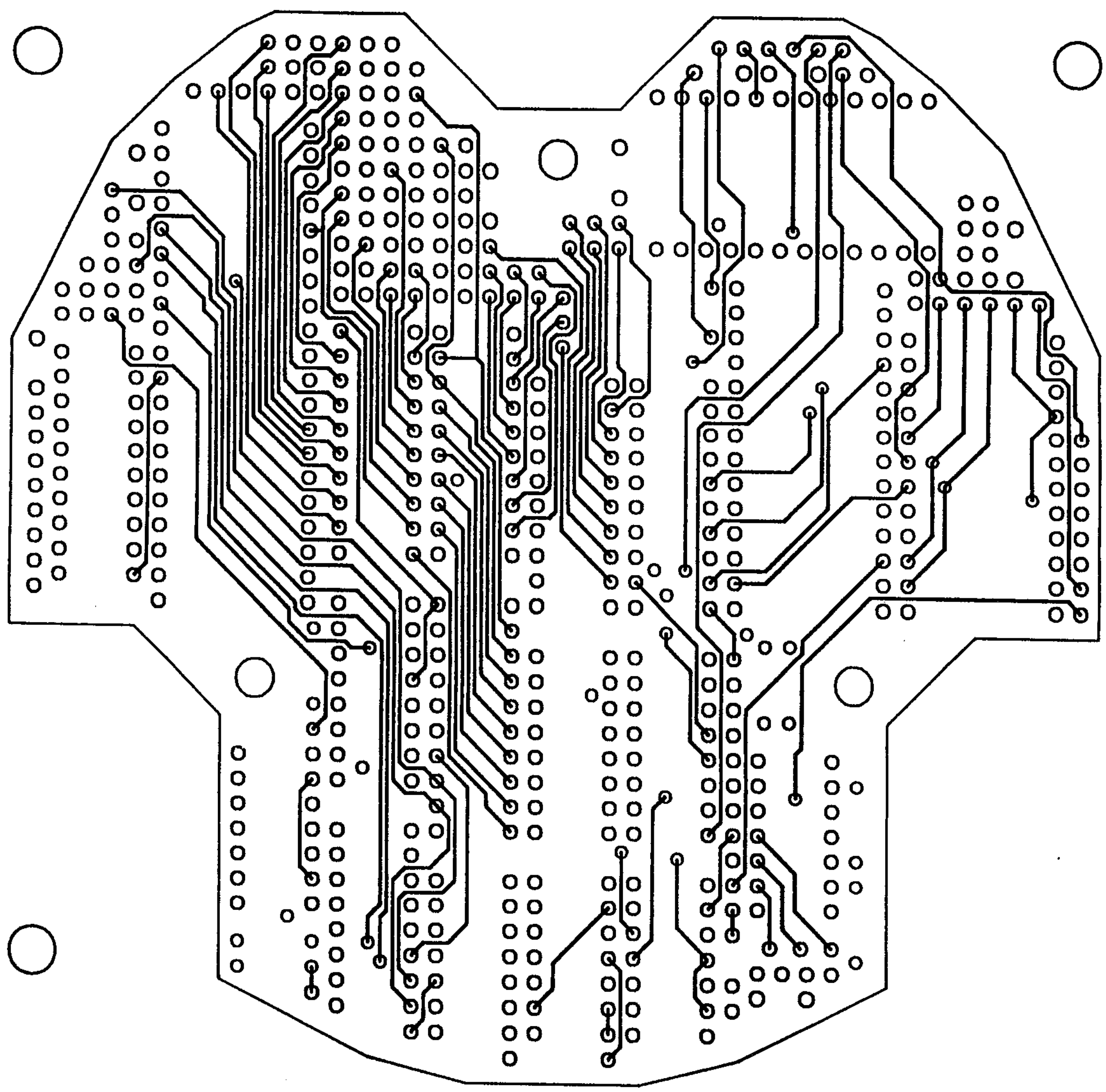
LAYER 2 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B 13016.020		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5   REV -   9/16/92
SIGNAL LAYER
LAYER 3 OF 6

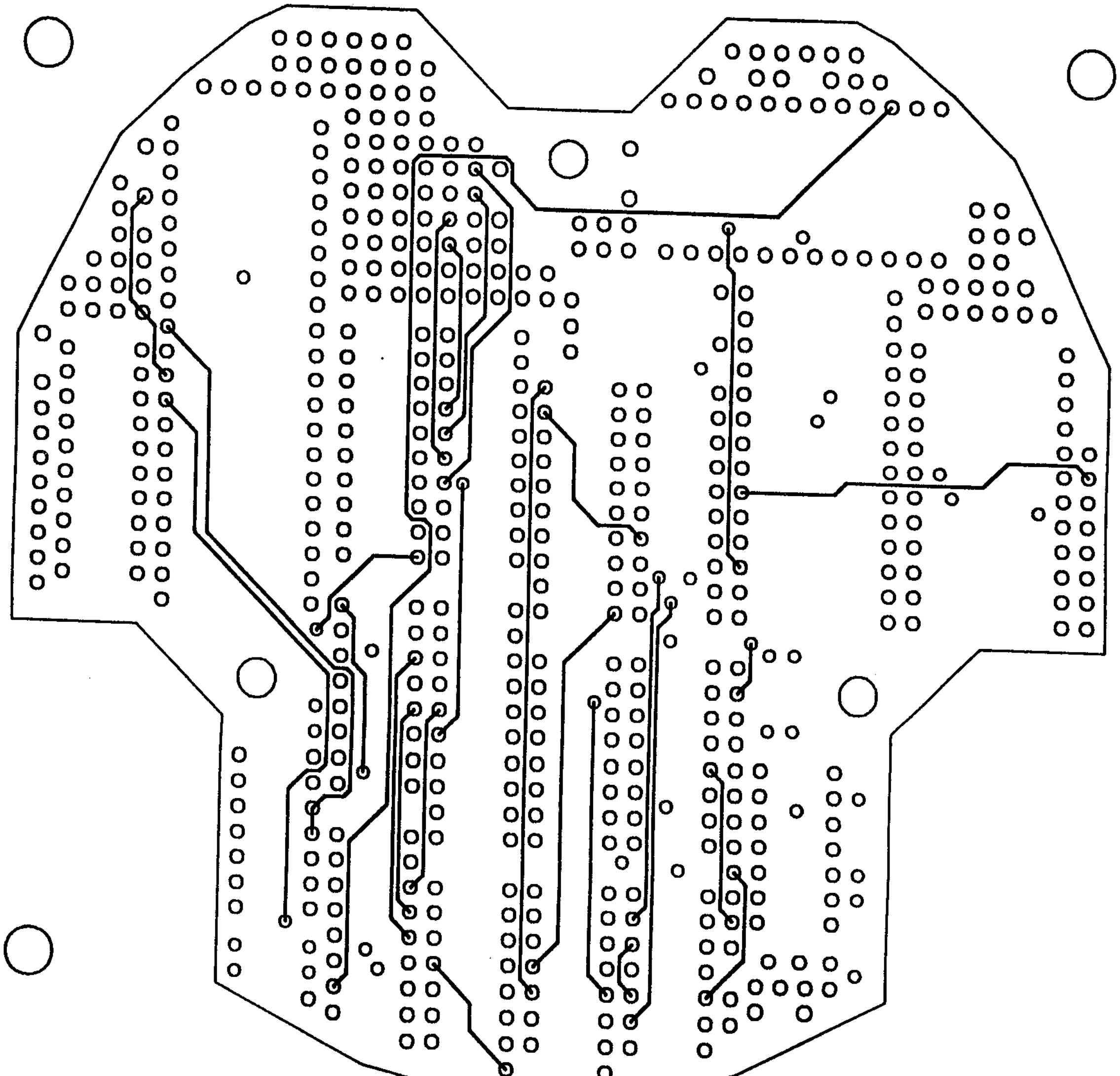


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.030		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92

SIGNAL LAYER
LAYER 4 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

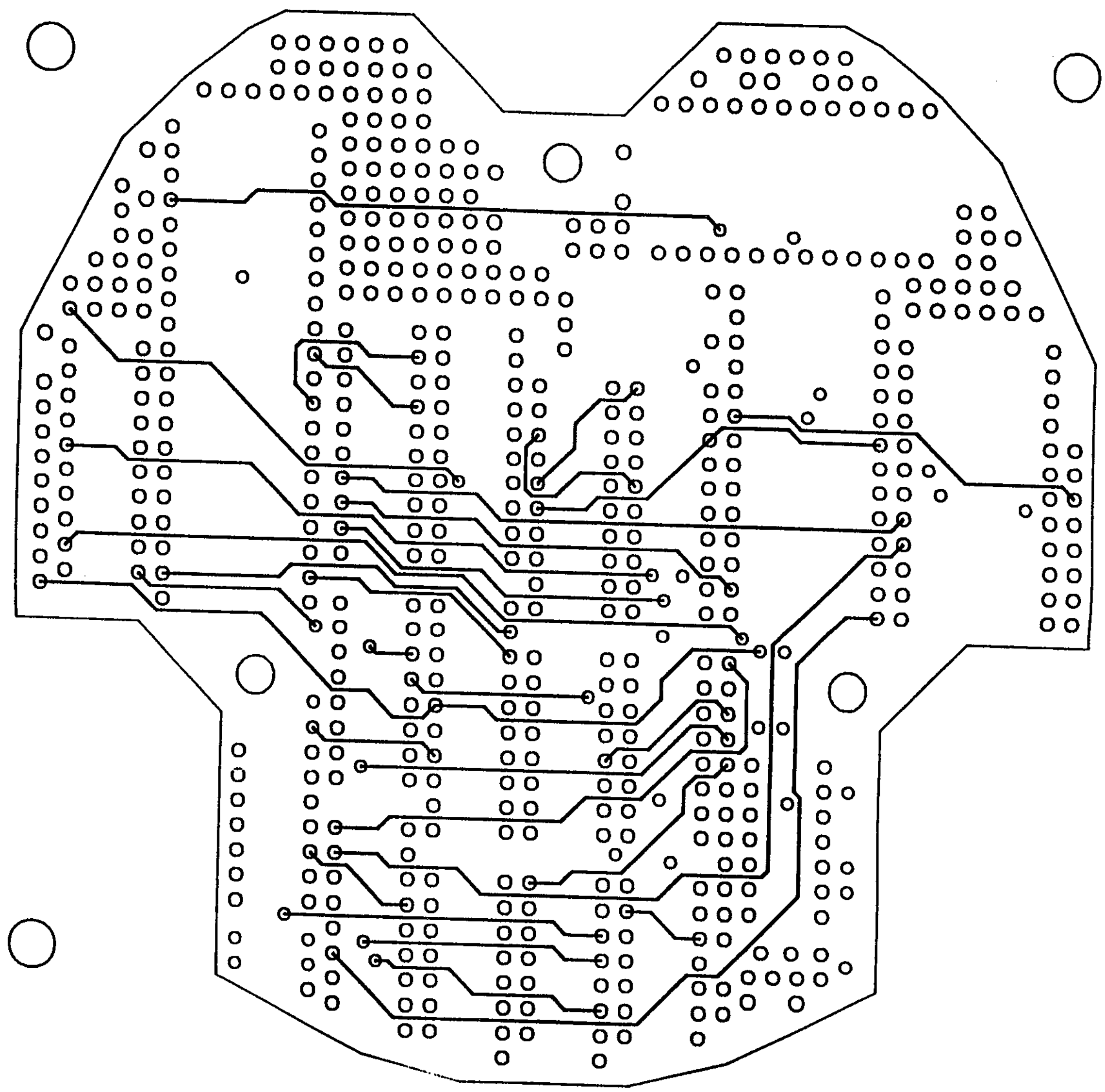
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.040		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92

SIGNAL LAYER

LAYER 5 OF 6

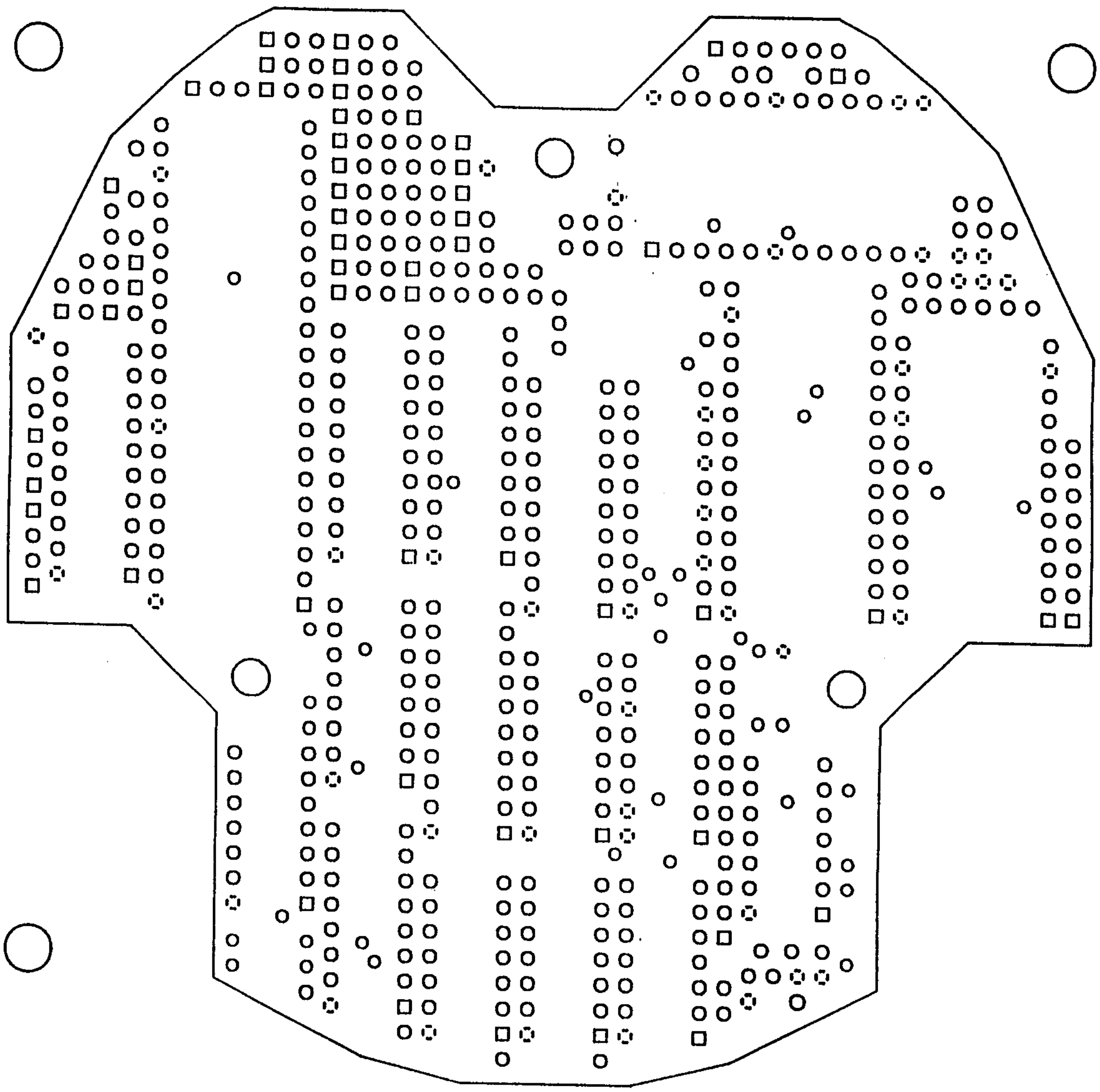


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.050		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92

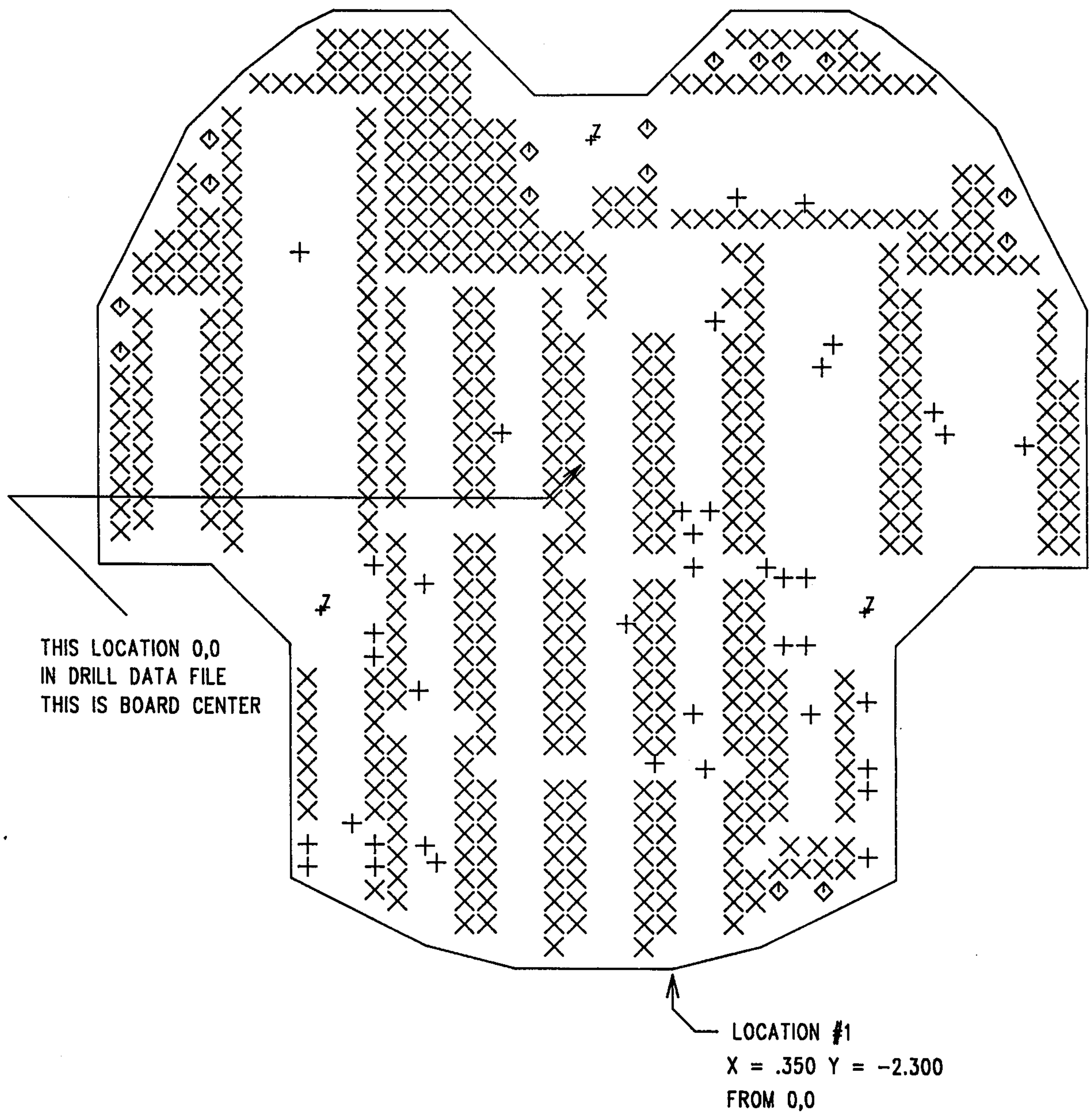
POWER PLANE
LAYER 6 OF 6



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.060		SHEET NO.

APPLIED RESEARCH CORPORATION DRILL DRAWING
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. D 13016.100		SHEET NO.

SIZE	QTY	SYM
28	40	+
32	537	X
37	16	◇
160	3	Z

FABRICATION NOTES:

1. FINISHED BOARD THICKNESS TO BE .062 +/- .007
2. MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
   TYPE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
3. FABRICATE IN ACCORDANCE WITH MIL-P-55110
4. ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
5. LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
   LAYER 1 -COMPONENT SIDE THRU LAYER 6 -SOLDER SIDE
6. MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
7. TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
8. ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
9. ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
   IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
   PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
10. SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
    PER MIL-I-43553
10. ALL .160" DIA. HOLES WILL BE NON-PLATED THRU (3PLCS.)

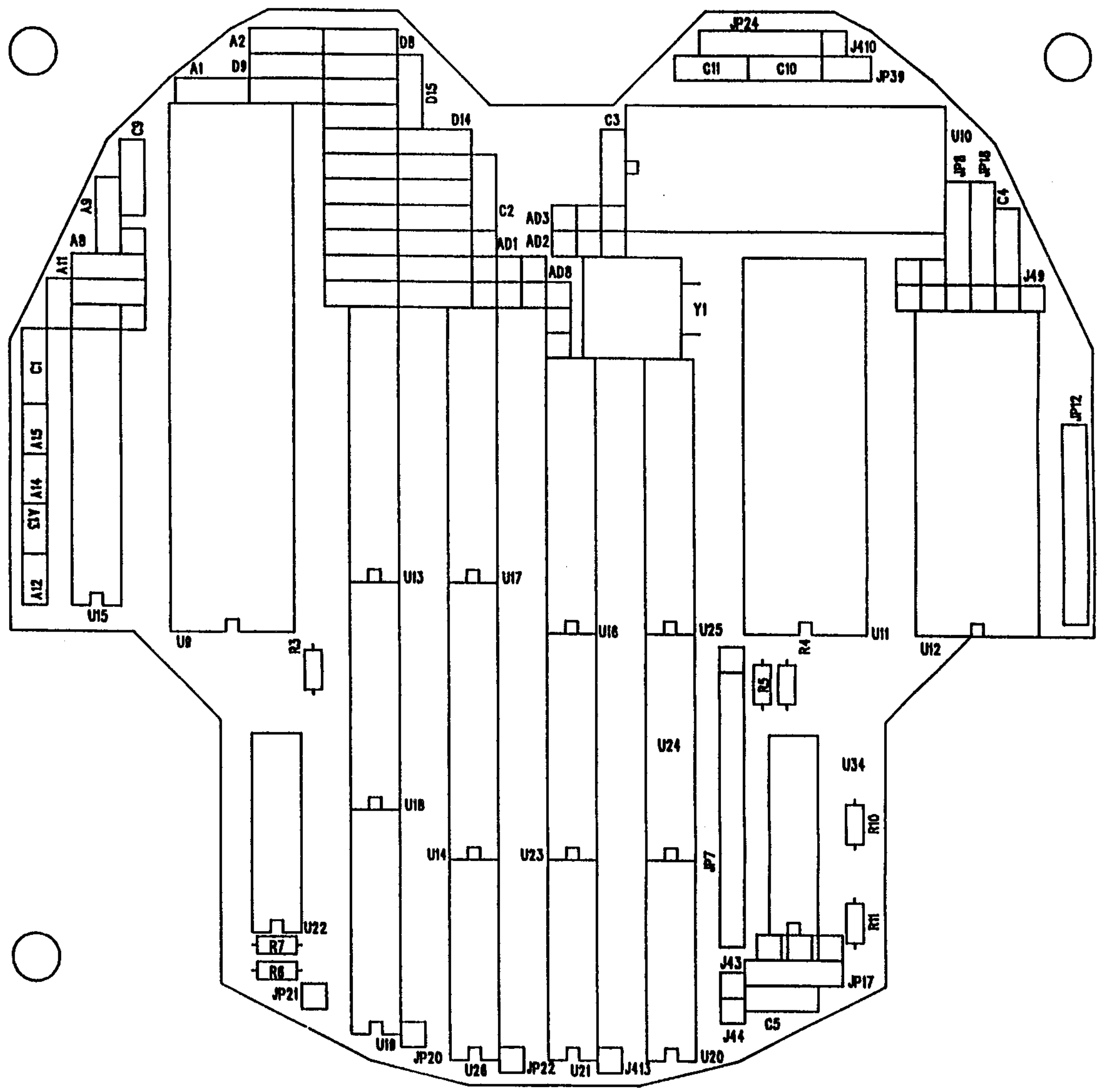
BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION #1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.700	1.450
6.	-2.050	1.800
7.	-2.550	1.800
8.	-2.550	2.950
9.	-2.150	3.750
10.	-1.900	4.000
11.	-1.650	4.200
12.	-1.500	4.275
13.	-.975	4.275
14.	-.600	3.900
15.	-.100	3.900
16.	.275	4.275
17.	.800	4.275
18.	.950	4.200
19.	1.200	4.000
20.	1.450	3.750
21.	1.850	2.950
22.	1.850	1.800
23.	1.350	1.800
24.	1.000	1.450
25.	1.000	.400
26.	.400	.100
27.	0	0

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

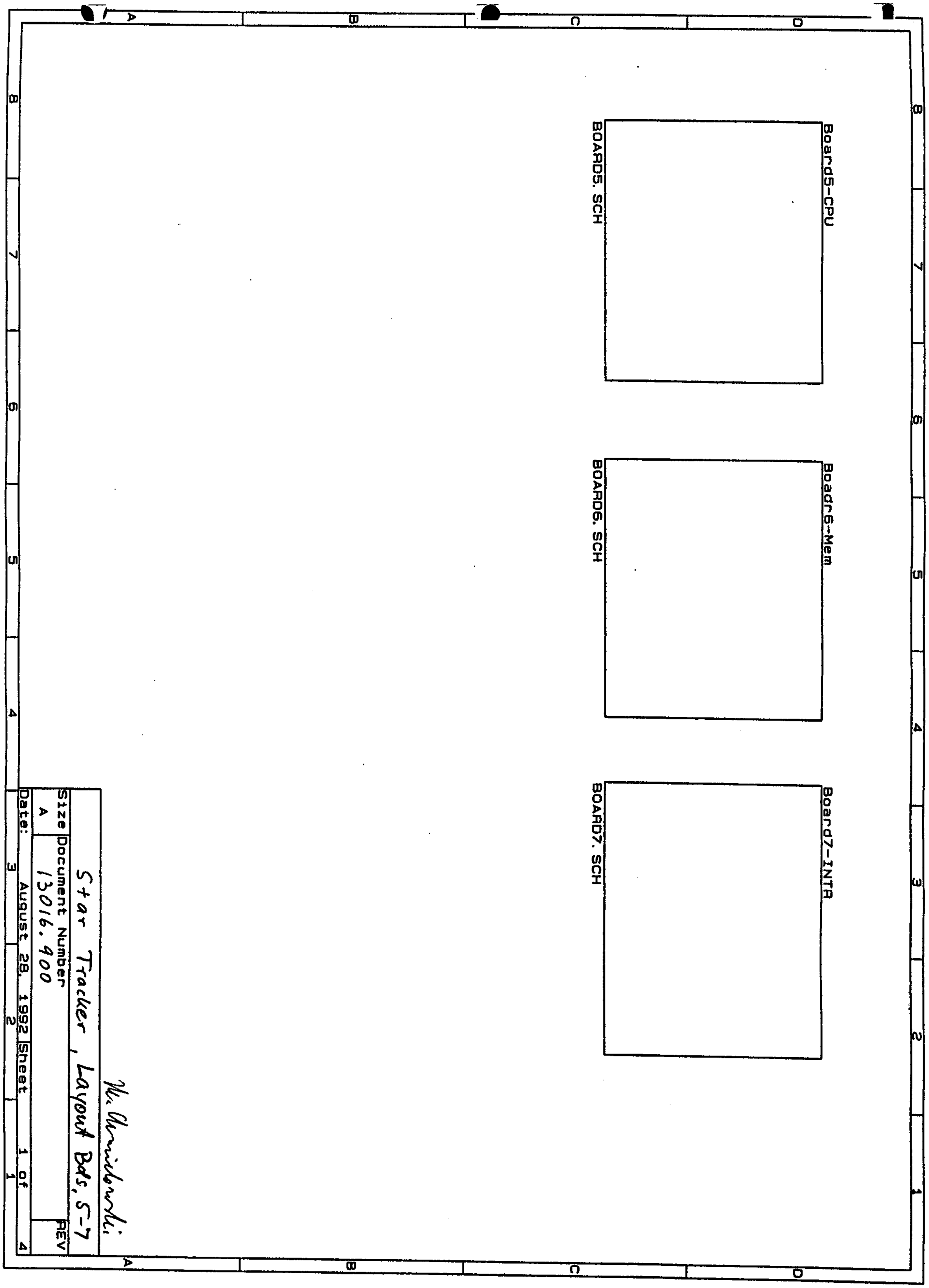
APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	
DRAWING NO. D 13016.10φ		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #5 REV - 9/16/92
SILKSCREEN
LAYER 1 OF 6

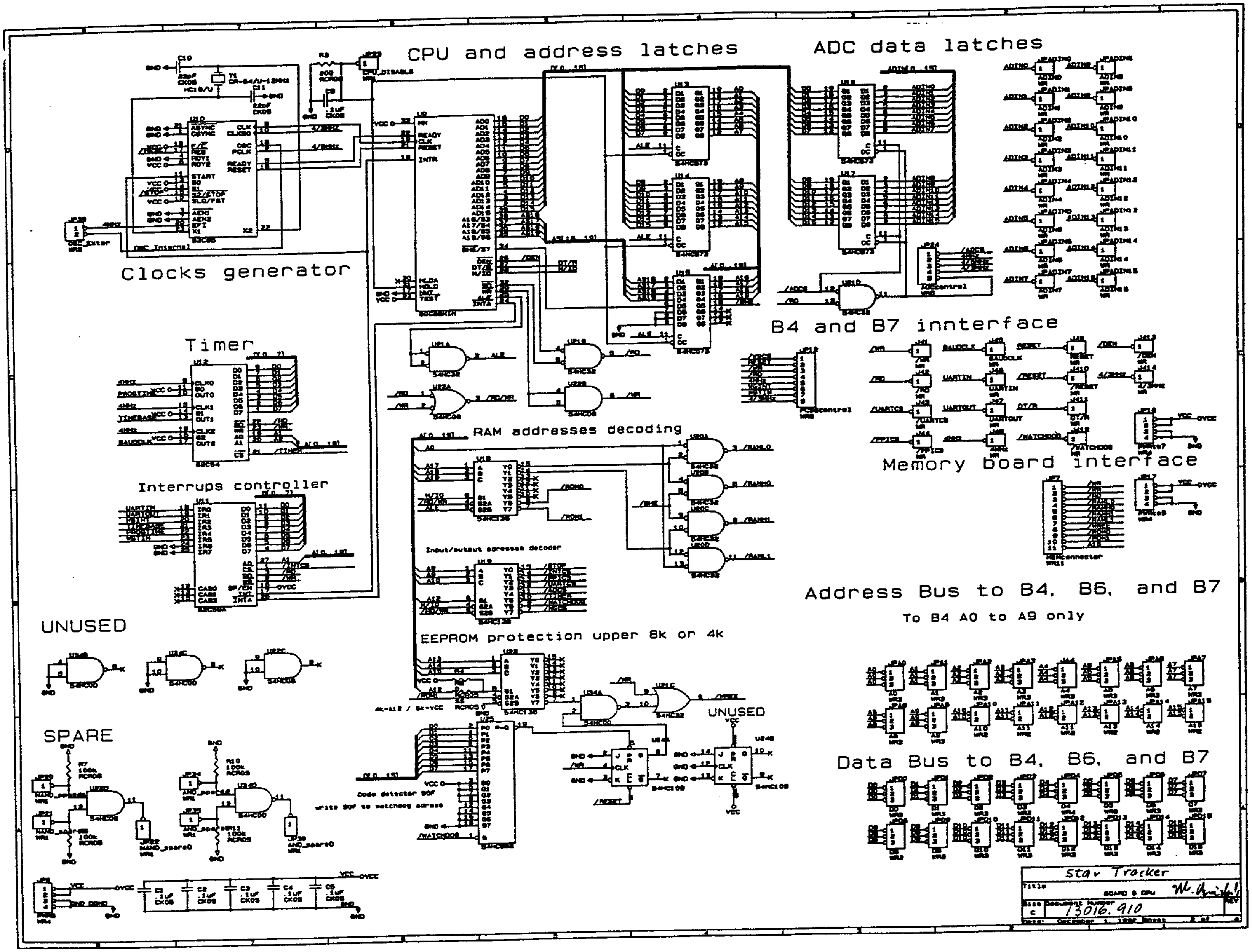


Use or disclosure of these
SBIR data is subject to the
restriction on the title page
of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE 9/16/92
DRAWING NO. B13016.110		SHEET NO.

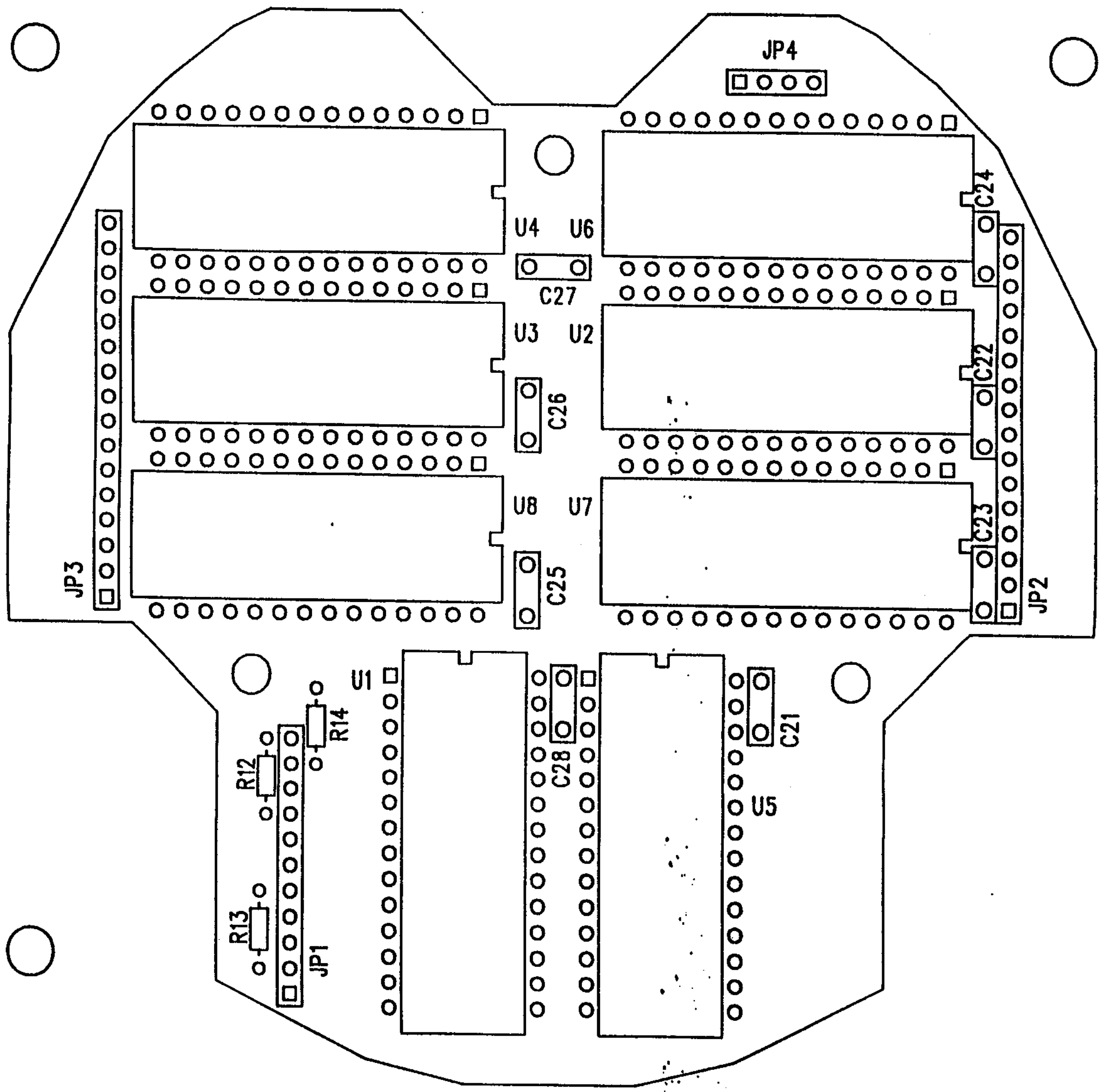


Use or disclosure of these
SBIR data is subject to the
restriction on the title page
of this report.


CPU and address latches
ADC data latches
Clocks generator
B4 and B7 innterface
Timer
RAM addresses decoding
Memory board interface
Interrups controller
Input/output adresses decoder
Address Bus to B4, B6, and B7
To B4 A0 to A9 only
UNUSED
EEPROM protection upper 8k or 4k
UNUSED
SPARE
Data Bus to B4, B6, and B7
Star Tracker
Title
BOARD 3 CPU
13016.910

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92
ASSEMBLY DRAWING

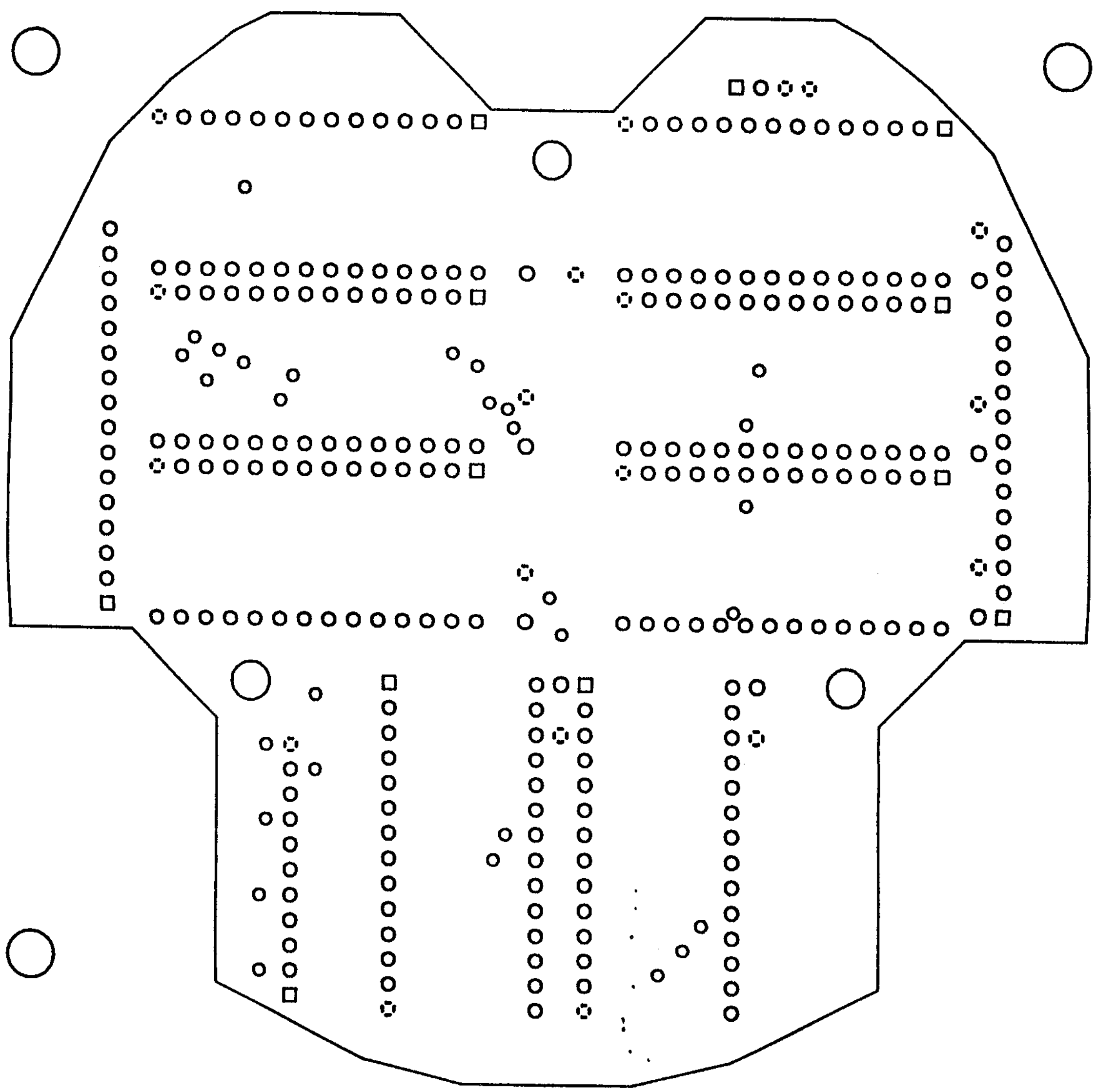
SILKSCREEN
LAYER 1 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13017.000		SHEET NO.

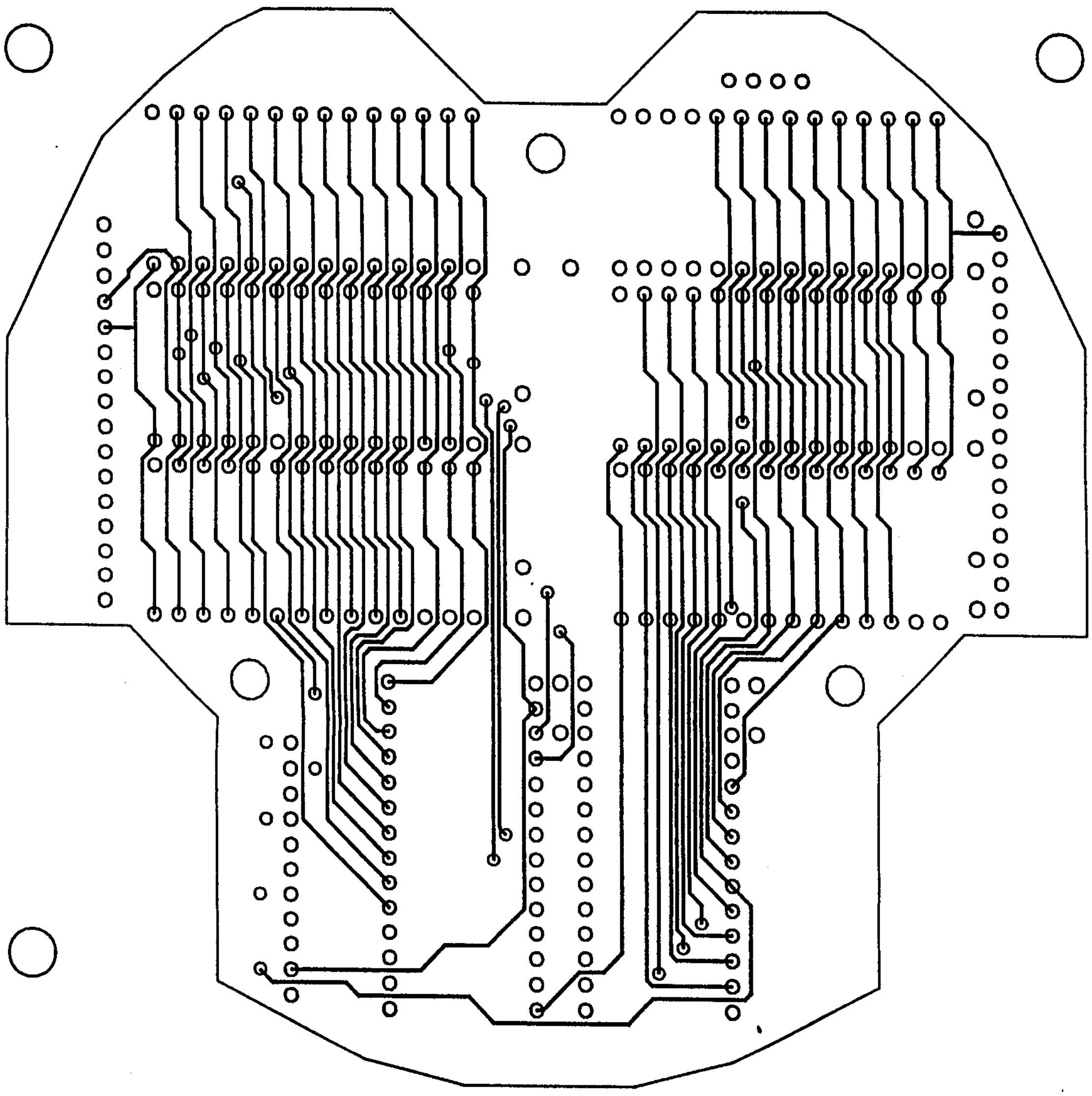
APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92
GROUND PLANE
LAYER 1 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13017.010		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92
SIGNAL LAYER
LAYER 2 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

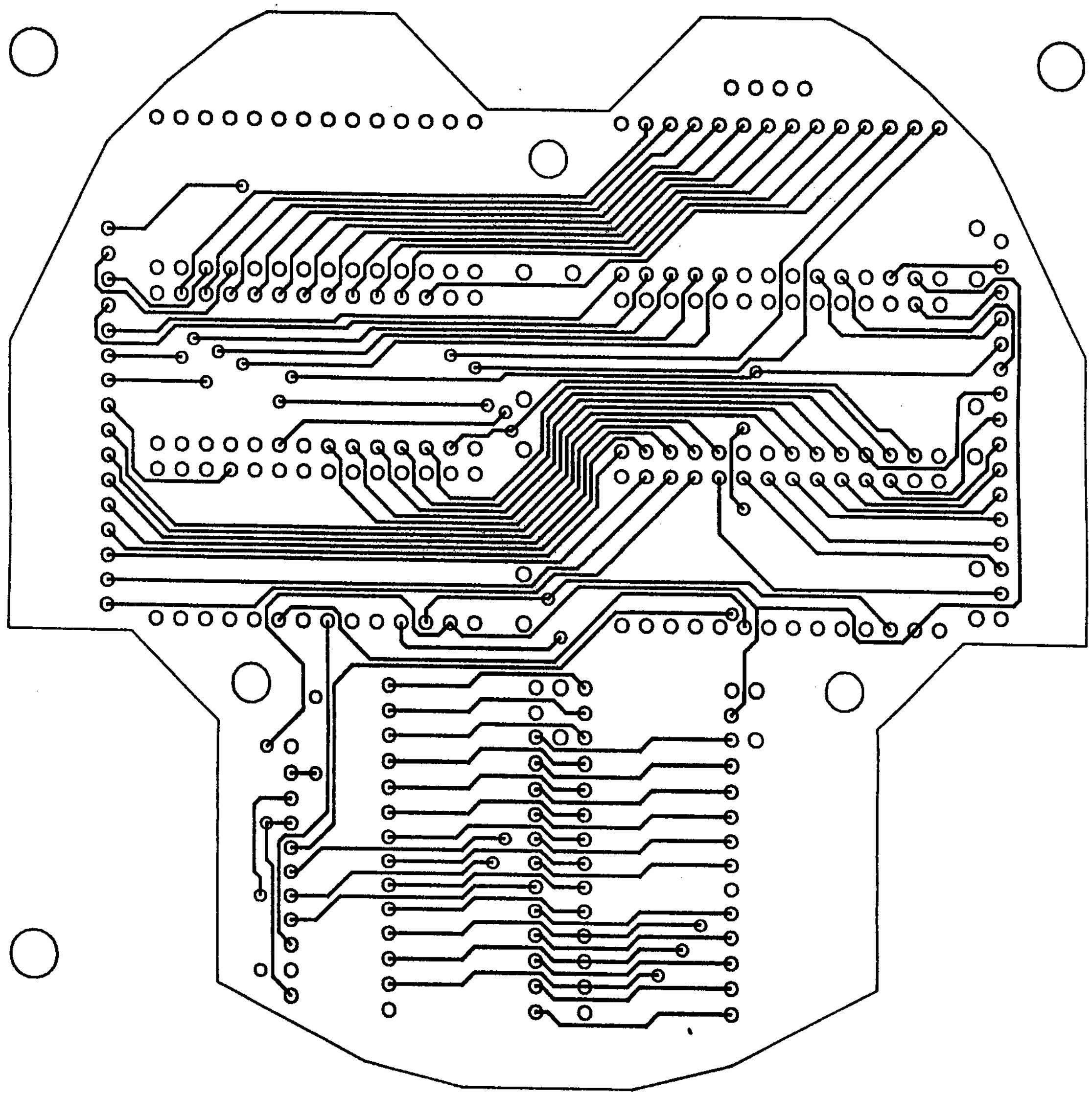
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13017.020		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92

SIGNAL LAYER

LAYER 3 OF 4



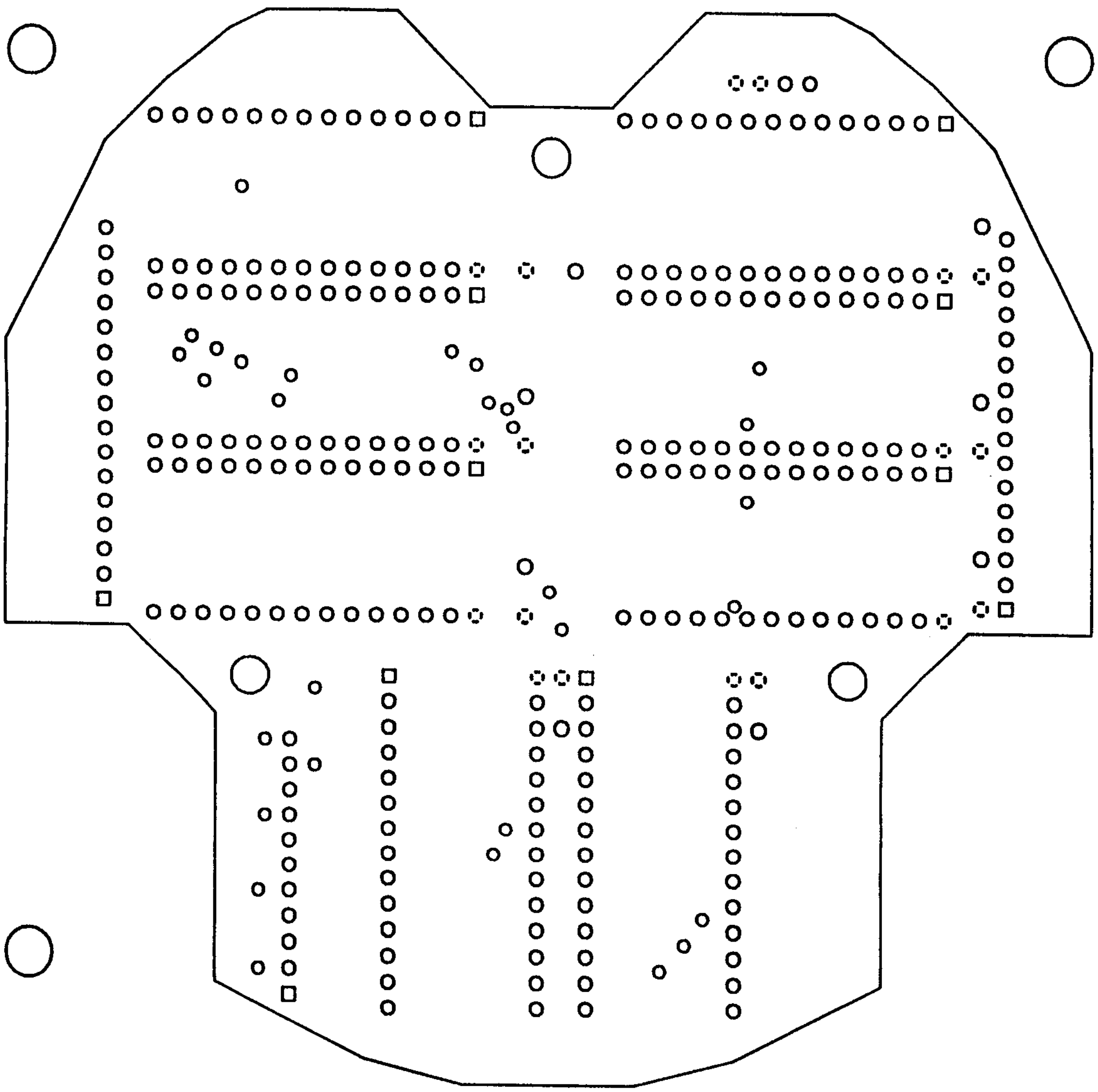
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13017.030		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92

POWER PLANE

LAYER 4 OF 4

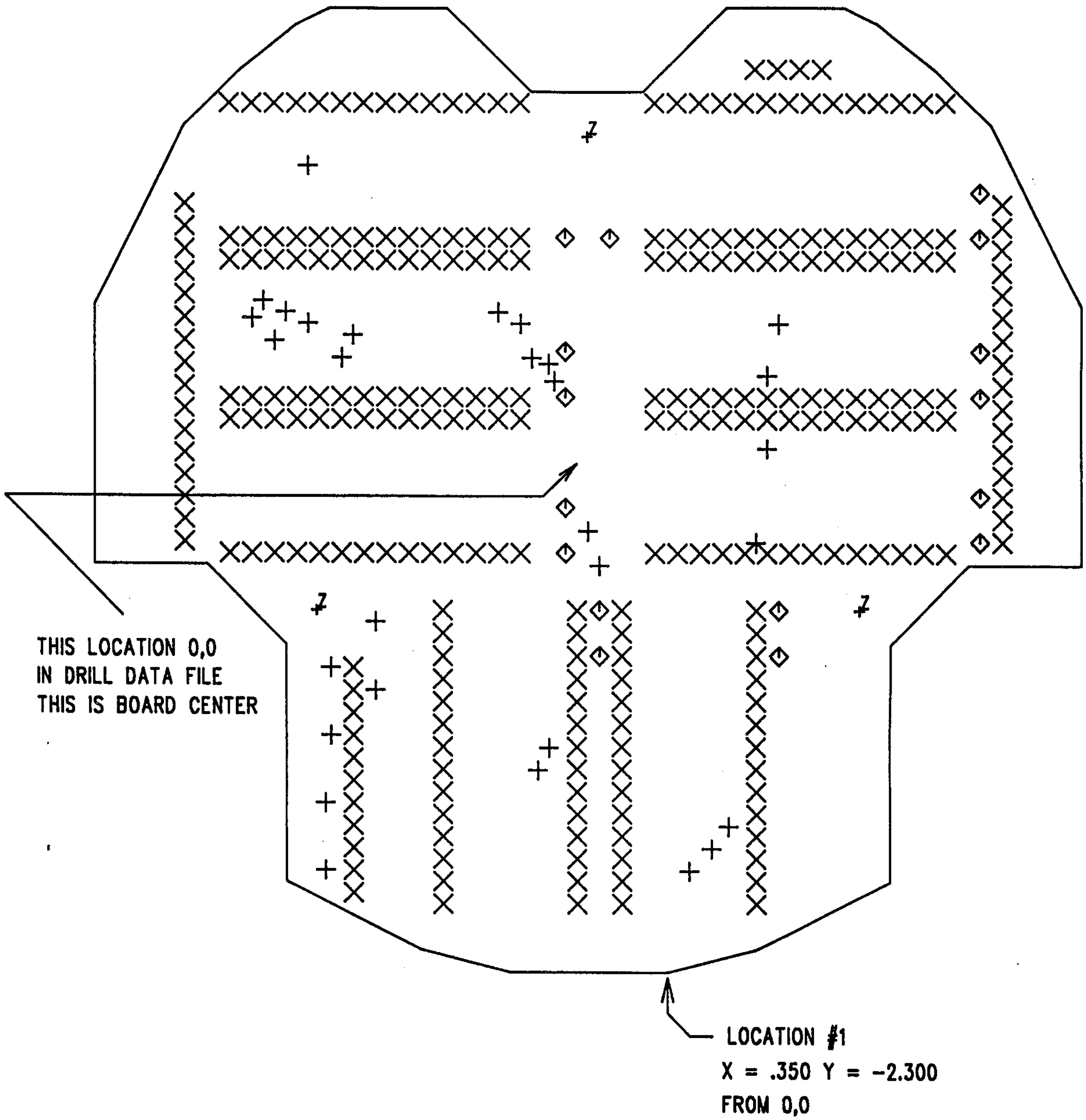


Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13017.040		SHEET NO.

APPLIED RESEARCH CORPORATION DRILL DRAWING

STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. D 13017.100		SHEET NO.

SIZE	QTY	SYM
28	30	+
32	271	×
37	16	◈
180	3	Z

BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION 1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.700	1.450
6.	-2.050	1.800
7.	-2.550	1.800
8.	-2.550	2.950
9.	-2.150	3.750
10.	-1.900	4.000
11.	-1.650	4.200
12.	-1.500	4.275
13.	-.975	4.275
14.	-.600	3.900
15.	-.100	3.900
16.	.275	4.275
17.	.800	4.275
18.	.950	4.200
19.	1.200	4.000
20.	1.450	3.750
21.	1.850	2.950
22.	1.850	1.800
23.	1.350	1.800
24.	1.000	1.450
25.	1.000	.400
26.	.400	.100
27.	0	0

FABRICATION NOTES:

1. FINISHED BOARD THICKNESS TO BE .062 +/- .007
2. MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
   TYPE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
3. FABRICATE IN ACCORDANCE WITH MIL-P-55110
4. ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
5. LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
   LAYER 1 -COMPONENT SIDE THRU LAYER 4 -SOLDER SIDE
6. MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
7. TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
8. ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
9. ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
   IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
   PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
10. SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
    PER MIL-I-43553
10. ALL .160" DIA. HOLES WILL BE NON-PLATED THRU (3PLCS.)

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

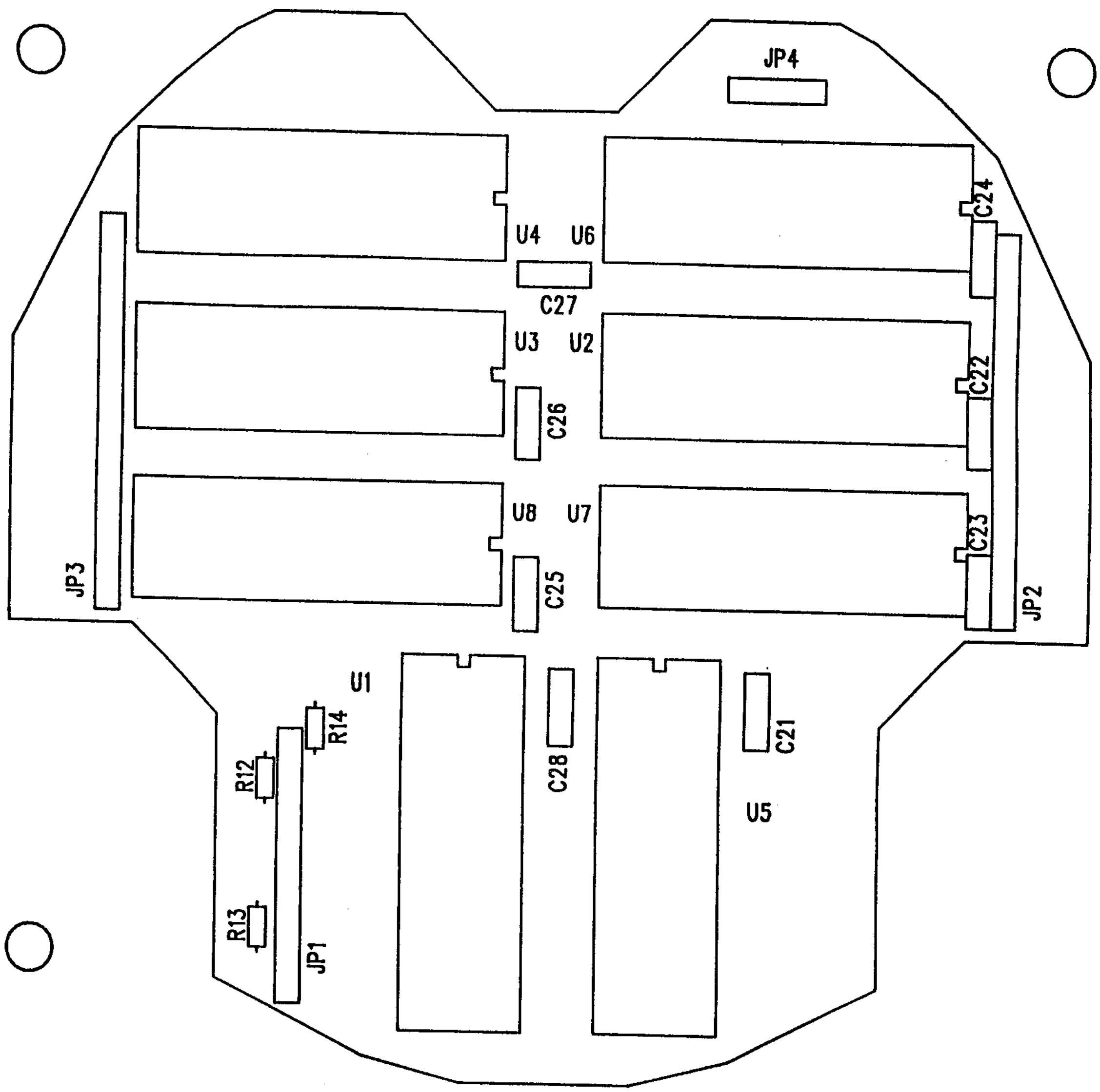
APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	
DRAWING NO. D 13017.100		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #6 REV - 9/13/92

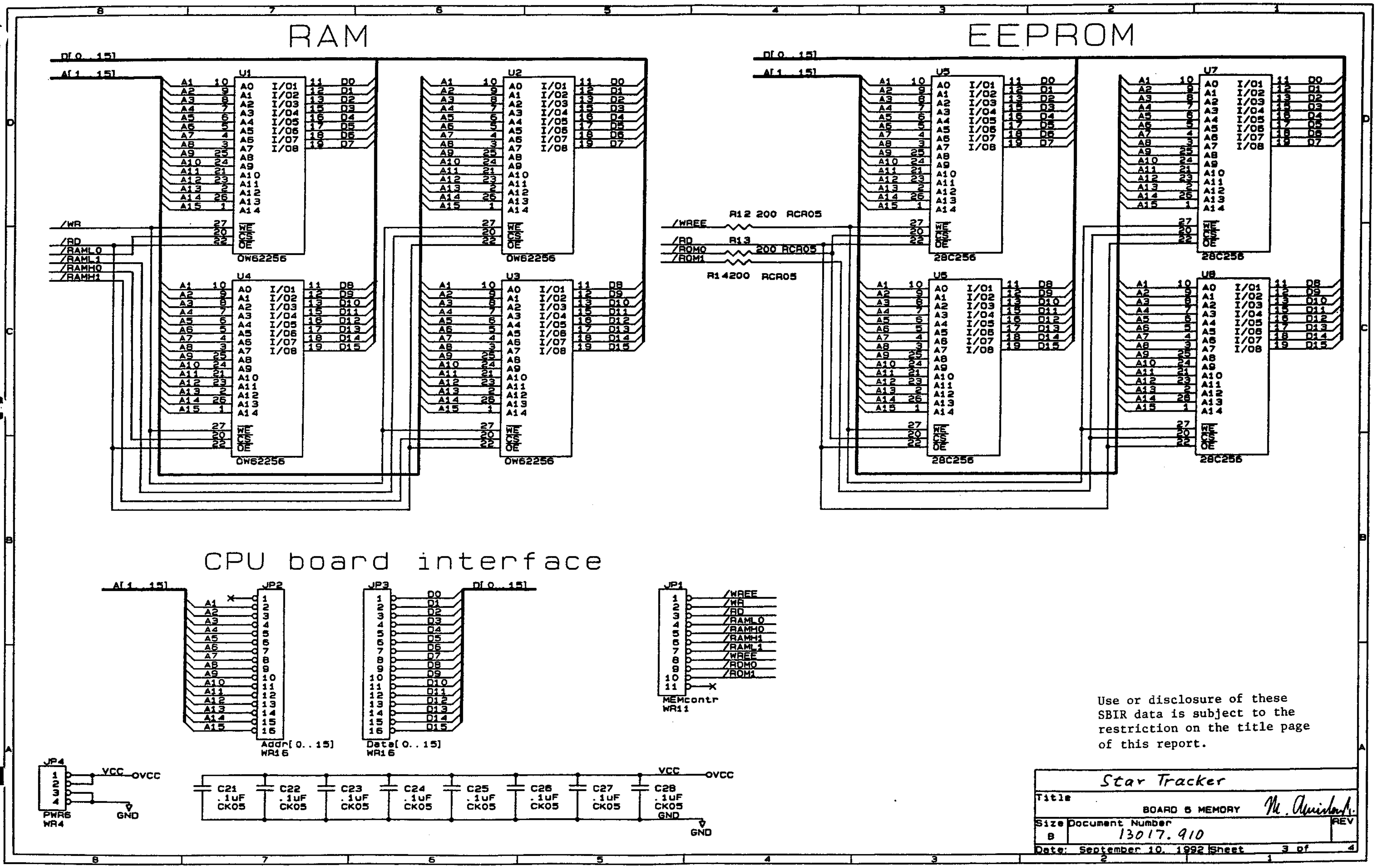
SILKSCREEN

LAYER 1 OF 4



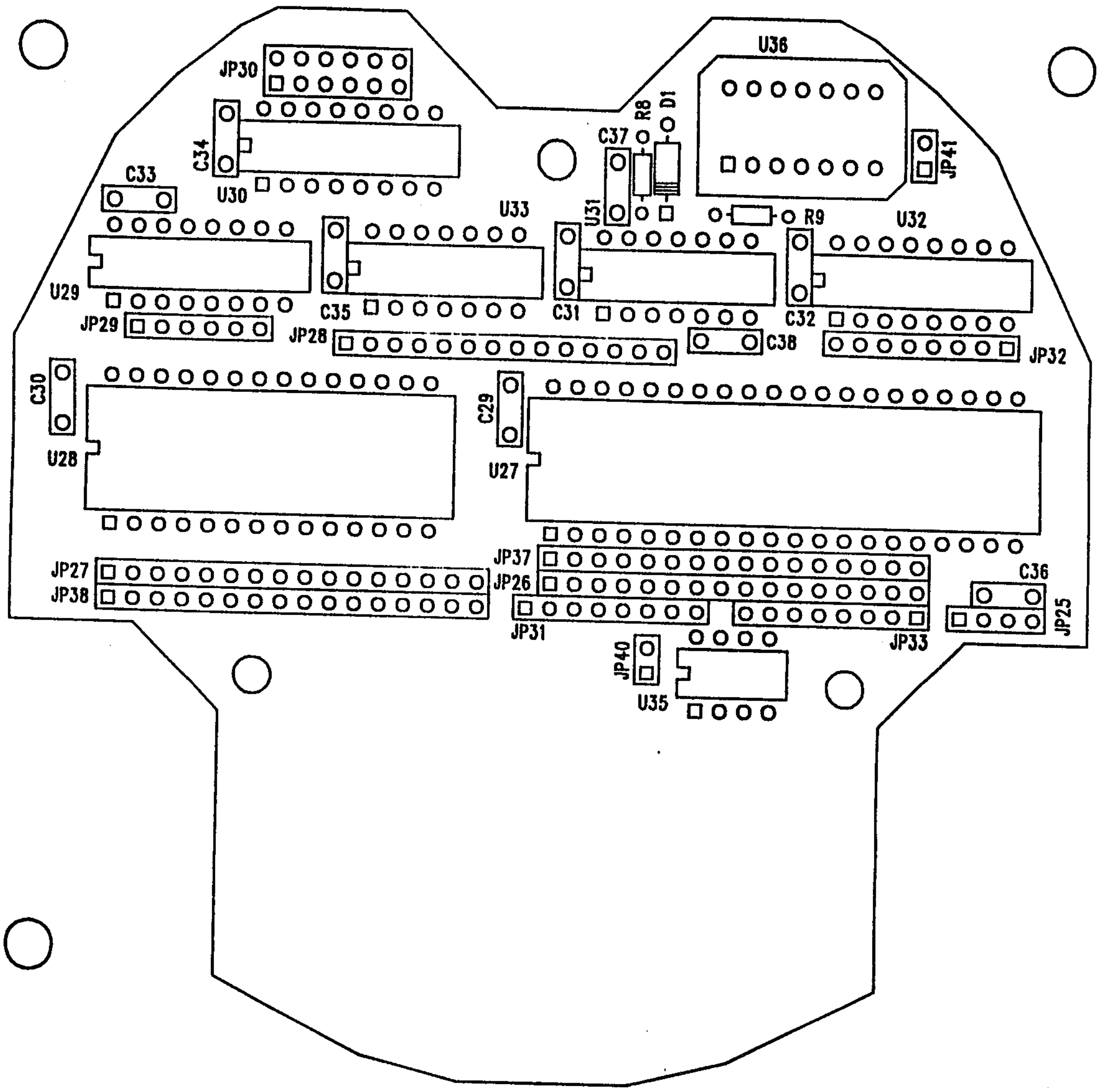
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13017.110		SHEET NO.



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92
ASSEMBLY DRAWING



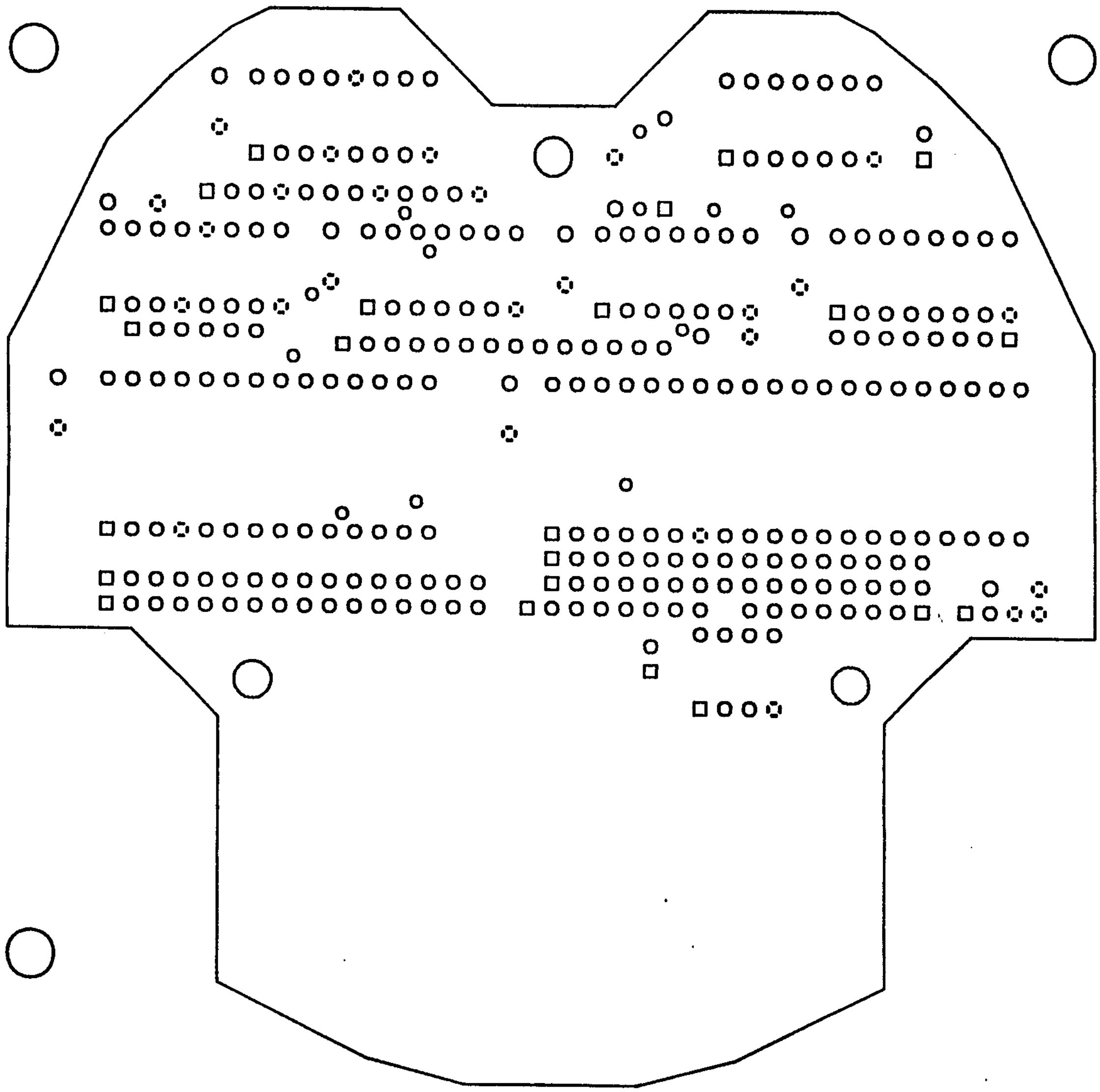
Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
B 13018.000	

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92
GROUND PLANE
LAYER 1 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

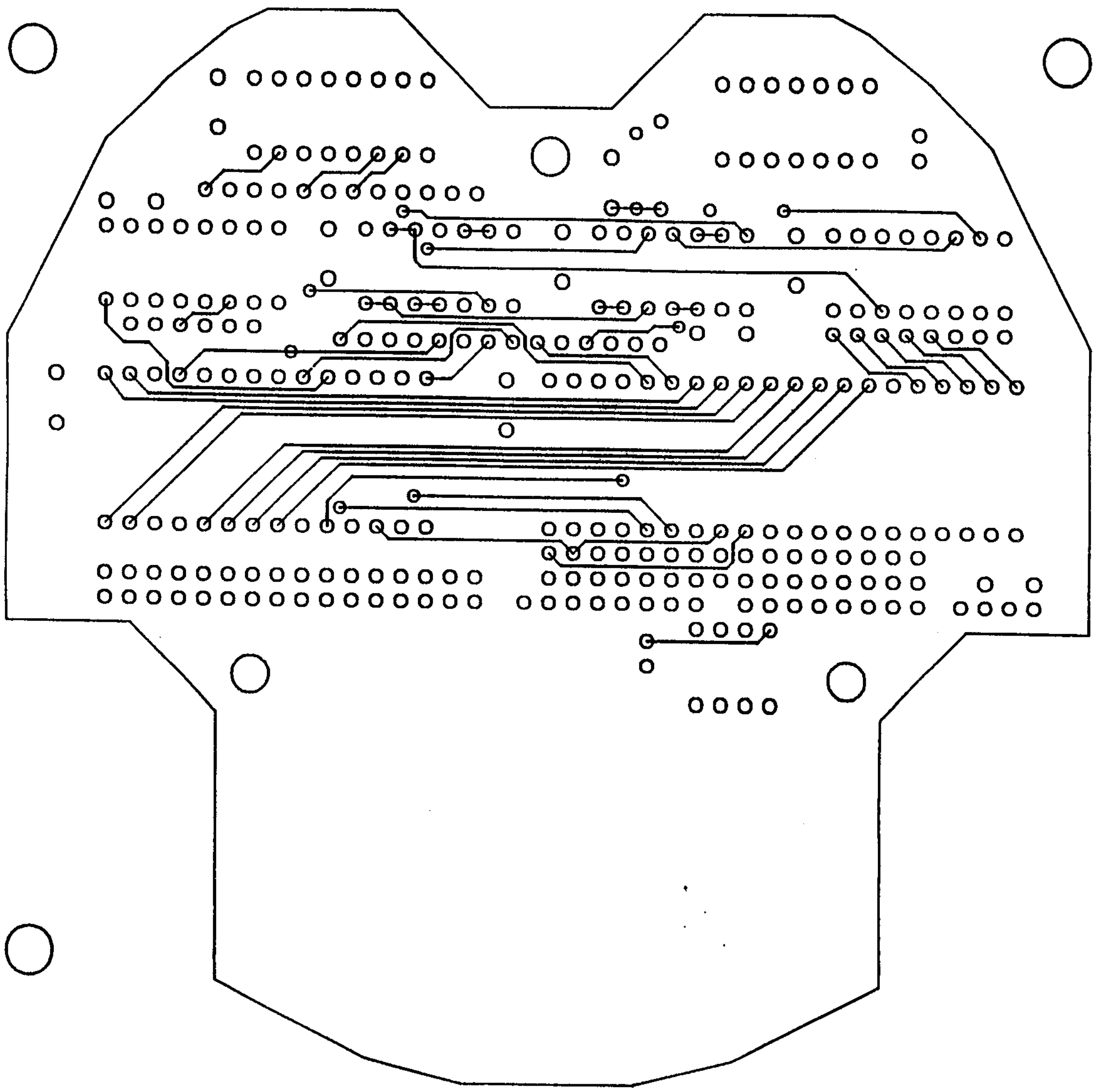
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13018.010		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92

SIGNAL LAYER

LAYER 2 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

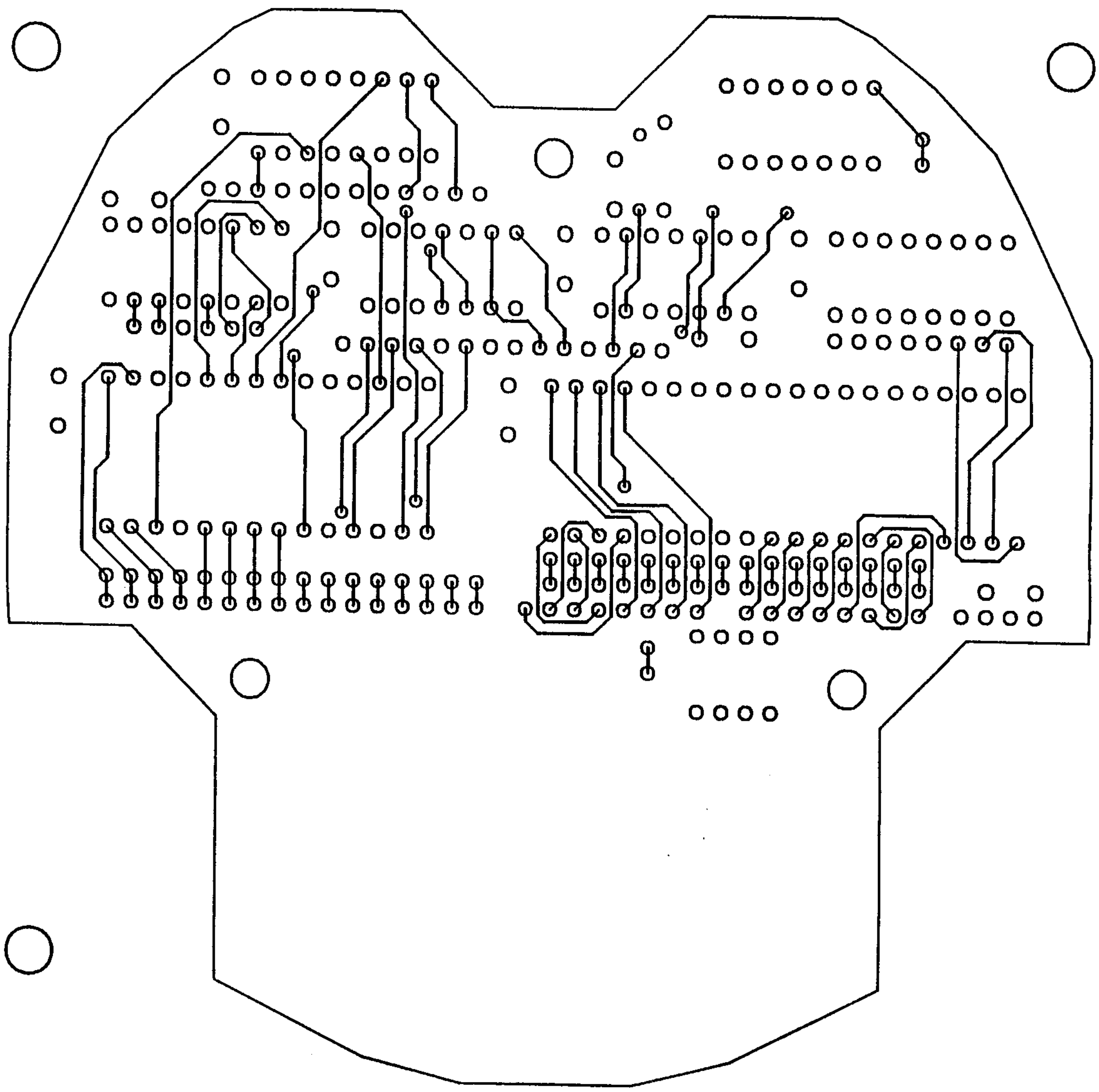
APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13018.020		SHEET NO.

APPLIED RESEARCH CORPORATION

STARTRACKER ELECTRONICS BOARD #7   REV -   9/13/92

SIGNAL LAYER

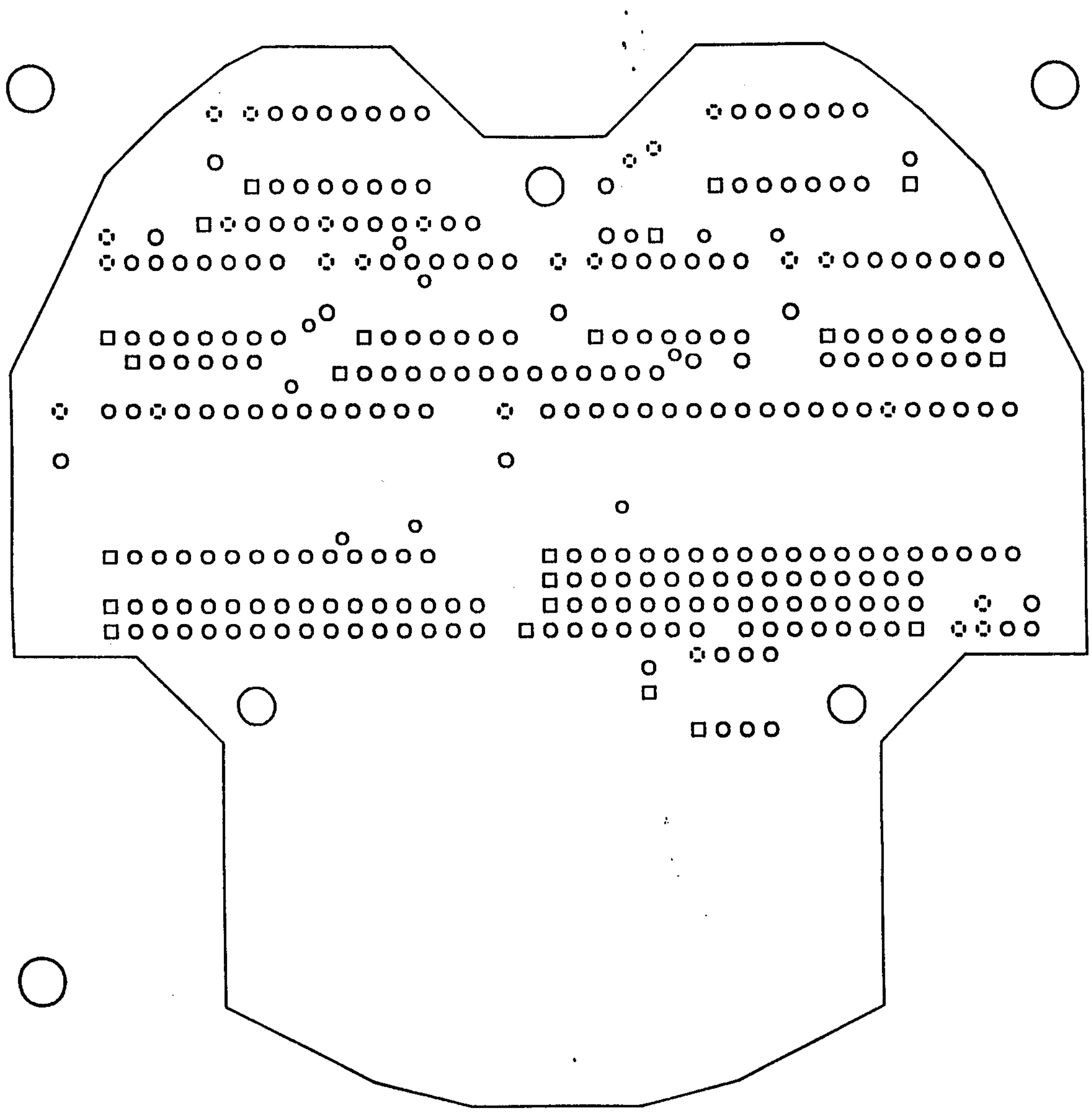
LAYER 3 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B 13018.030		SHEET NO.

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92
POWER PLANE
LAYER 4 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

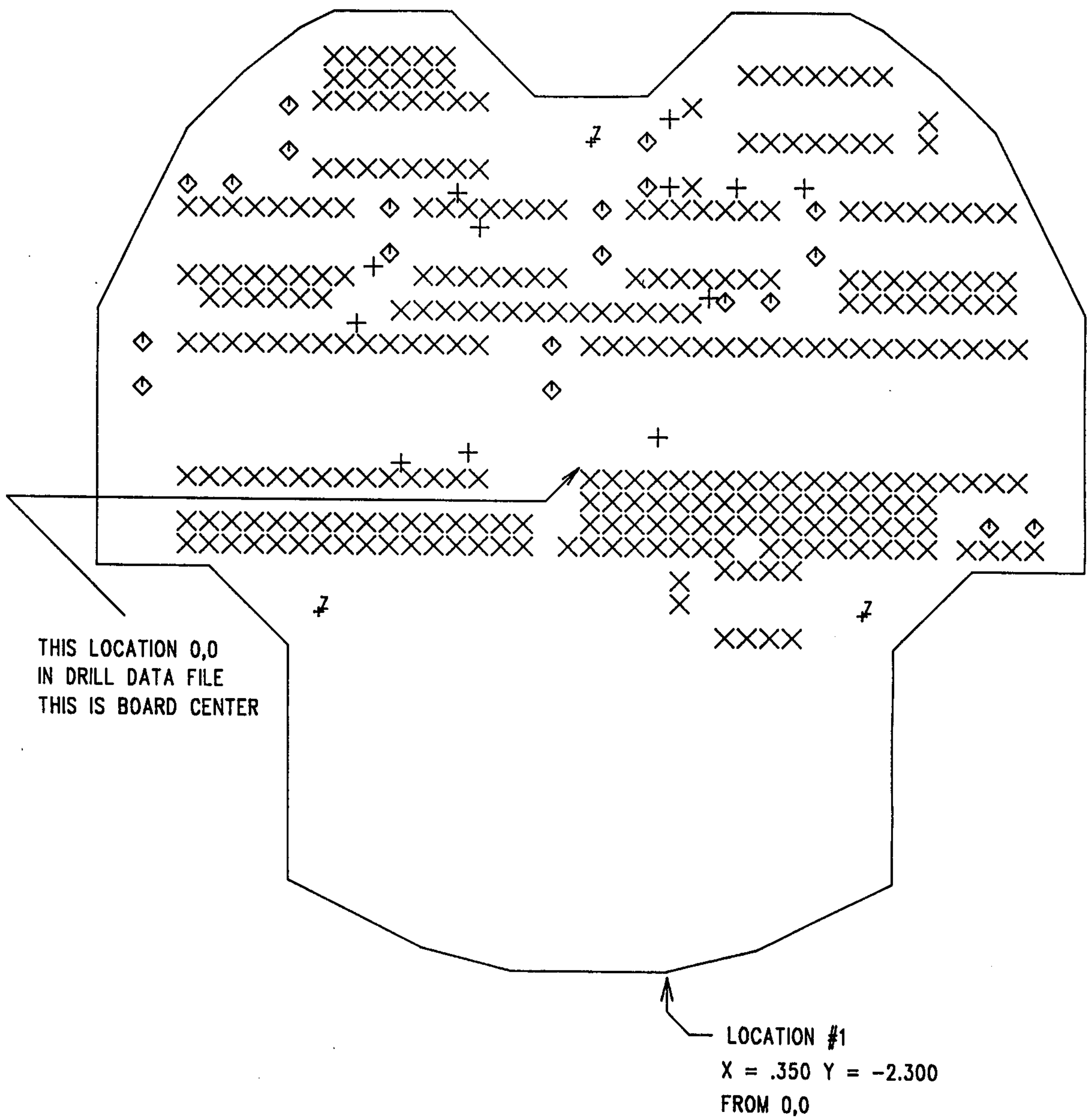
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
B13018.040	

APPLIED RESEARCH CORPORATION DRILL DRAWING

STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92



APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. D 13018.100		SHEET NO.

SIZE	QTY	SYM
28	12	+
32	296	X
37	20	◇
160	3	Z

BOARD OUTLINE COORDINATES
CLOCKWISE FROM LOCATION #1

	X	Y
1.	0	0
2.	-.700	0
3.	-1.100	.100
4.	-1.700	.400
5.	-1.700	1.450
6.	-2.050	1.800
7.	-2.550	1.800
8.	-2.550	2.950
9.	-2.150	3.750
10.	-1.900	4.000
11.	-1.650	4.200
12.	-1.500	4.275
13.	-.975	4.275
14.	-.600	3.900
15.	-.100	3.900
16.	.275	4.275
17.	.800	4.275
18.	.950	4.200
19.	1.200	4.000
20.	1.450	3.750
21.	1.850	2.950
22.	1.850	1.800
23.	1.350	1.800
24.	1.000	1.450
25.	1.000	.400
26.	.400	.100
27.	0	0

FABRICATION NOTES:

1. FINISHED BOARD THICKNESS TO BE .062 +/- .007
2. MATERIAL TYPE GFN-XXX-C1/C1-A-2-C PER MIL-P-13949
   TYPE GFN-XXX-CA/CA-A-2-C IS ACCEPTABLE FOR OUTER LAYERS
3. FABRICATE IN ACCORDANCE WITH MIL-P-55110
4. ALL MASTER ARTWORKS ARE VIEWED THRU COMPONENT SIDE
5. LAY-UP OF LAYERS TO BE AS MARKED ON MASTER ARTWORKS
   LAYER 1 -COMPONENT SIDE THRU LAYER 4 -SOLDER SIDE
6. MINIMUM CONDUCTOR WIDTH AFTER ETCH IS .010"
7. TIN/LEAD REFLOW CONDUCTOR PATTERN PER MIL-275
8. ALL DIMENSIONS ARE IN INCHES TOLERANCE +/- .005 UNLESS OTHERWISE NOTED
9. ALL HOLE SIZES LISTED IN THE DRILL SYMBOL CHART ARE
   IN THOUSANDTHS OF AN INCH AND ARE FINISHED DIAMETERS AFTER
   PLATING. HOLE TOLERANCE +/- .003 UNLESS OTHERWISE NOTED
10. SILKSCREEN TO BE APPLIED TO COMPONENT SIDE COLOR WHITE
    PER MIL-I-43553
10. ALL .160" DIA. HOLES WILL BE NON-PLATED THRU (3PLCS.)

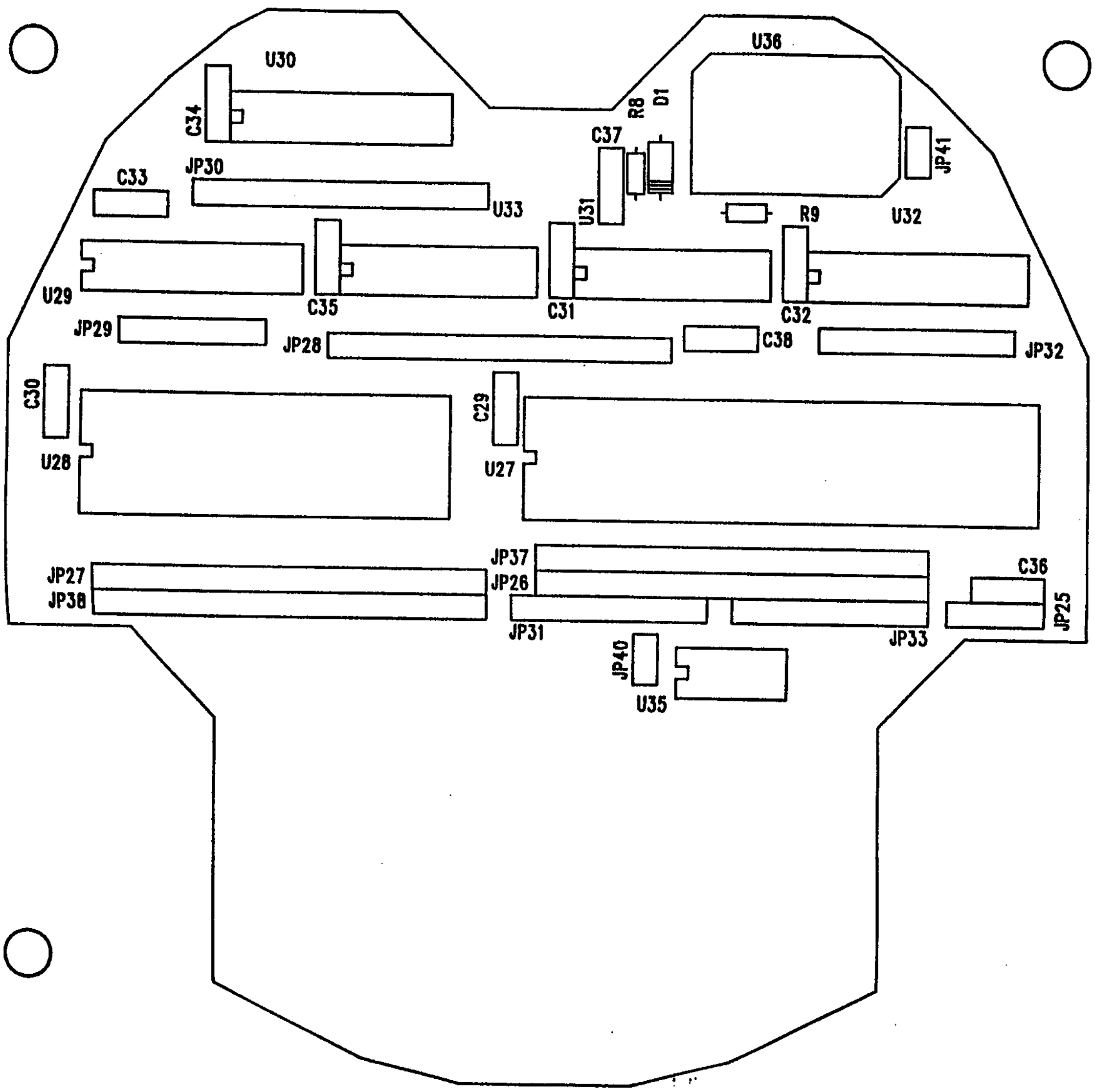
APPLIED RESEARCH CORP.
8201 CORPORATE DR., LANDOVER, MD 20785

APPROVED BY	DRAWN BY	DATE
Siegfried Auer	Phil Goodwin	

DRAWING NO.	SHEET NO.
D 13018.10[illegible]	

APPLIED RESEARCH CORPORATION
STARTRACKER ELECTRONICS BOARD #7 REV - 9/13/92

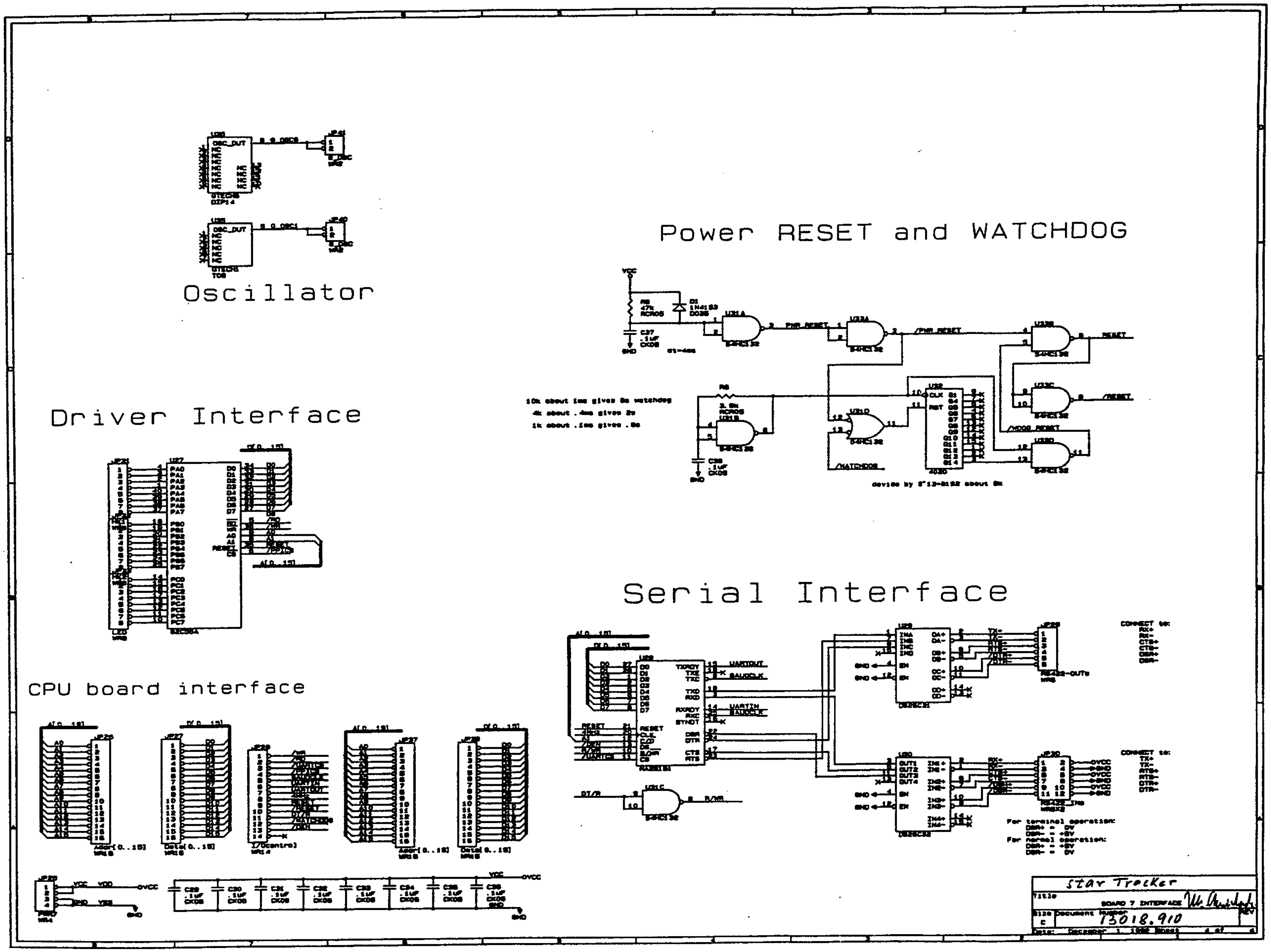
SILKSCREEN
LAYER 1 OF 4



Use or disclosure of these SBIR data is subject to the restriction on the title page of this report.

APPLIED RESEARCH CORP. 8201 CORPORATE DR., LANDOVER, MD 20785		
APPROVED BY Siegfried Auer	DRAWN BY Phil Goodwin	DATE
DRAWING NO. B13018.110		SHEET NO.

2.


Oscillator
Power RESET and WATCHDOG
Driver Interface
Serial Interface
CPU board interface
Star Tracker
BOARD 7 INTERFACE
13018.910

PARTS LIST

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #1	PL# B13012.200	Rev. 0	Sheet 1 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C1	M39014/01-1594	CAP,FXD,CER,0.1µF, 20%, 50V, REL=S			47
	C3	"	"			
	C4	"	"			
	C5	"	"			
	C9	"	"			
	C10	"	"			
	C11	"	"			
	C13	"	"			
	C14	"	"			
	C17	"	"			
	C18	"	"			
	C19	"	"			
	C20	"	"			
	C21	"	"			
	C23	"	"			
	C24	"	"			
	C27	"	"			
	C28	"	"			
	C29	"	"			
	C30	"	"			
	C31	"	"			
	C33	"	"			
	C34	"	"			
	C37	"	"			
	C38	"	"			
	C39	"	"			

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #1	PL# B13012.200	Rev. 0	Sheet 2 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C40	M39014/01-1594	CAP,FXD,CER,0.1µF,20%,50V,REL=S			
	C41	"	"			
	C42	"	"			
	C43	"	"			
	C44	"	"			
	C45	"	"			
	C46	"	"			
	C57	"	"			
	C58	"	"			
	C59	"	"			
	C78	"	"			
	C79	"	"			
	C80	"	"			
	C81	"	"			
	C82	"	"			
	C83	"	"			
	C84	"	"			
	C85	"	"			
	C86	"	"			
	C87	"	"			
	C88	"	"			
	C2	CCR05CH100FR	CAP,FXD,CER,TEMP-COMP,10pF,1%,200V,REL=R			1
	C6	M39014/01-1357	CAP,FXD,CER,1000pF,10%,200V,REL=S			2
	C7	"	"			
	C8	CCR05CJ4R7DR	CAP,FXD,CER,TEMP-COMP,4.7pF±0.5pF,200V,REL=R			1
	C12	CCR05CG330JR	CAP,FXD,CER,TEMP-COMP,33pF,5%,200V,REL=R			3

APPLIED RESEARCH CORPORATION	Title: Star Tracker PCB #1	PL#: B13012.200	Rev.: 0	Sheet 3 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C22	CCR05CG330JR	CAP,FXD,CER,TEMP-COMP,33pF,5%,200V,REL=R			
	C32	"	"			
	C15	M39014/01-1541	CAP,FXD,CER,0.022μF,10%,50V,REL=R			6
	C16	M39014/01-1541	CAP,FXD,CER,0.022μF,10%,50V			
	C25	"	"			
	C26	"	"			
	C35	"	"			
	C36	"	"			
	C89	M39003/01-K2770J	CAP,ELCTLT,15μF,20%,20V,REL=R			1
	L1	M39010/06B1R0KR	INDUCTOR, 1μH,10% REL=R	NYT		6
	L2	"	"	"		
	L4	"	"	"		
	L5	"	"	"		
	L7	"	"	"		
	L8	"	"	"		
	Q1	JANTXV2N918	TRANSISTOR,NPN	RAY		1
	Q2	FRL9130D3	MOSFET,P-CH,0.55 OHM	HAR		3
	Q4	"	"			
	Q6	"	"			
	Q3	FRL130D3	MOSFET,N-CH,0.18 OHM	HAR		4
	Q5	"	"			
	Q7	"	"			
	Q8	"	"			

APPLIED RESEARCH CORPORATION	Title Star Tracker	PL# B13012.200	Rev. 0	Sheet 4 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	D1	JV1N5711	diode, Schottky	HP		1
	D2	JV1N4148-1	diode	ITT		18
	D3	"	"			
	D4	"	"			
	D5	"	"			
	D6	"	"			
	D7	"	"			
	D8	"	"			
	D9	"	"			
	D10	"	"			
	D11	"	"			
	D12	"	"			
	D13	"	"			
	D14	"	"			
	D15	"	"			
	D16	"	"			
	D17	"	"			
	D22	"	"			
	D23	"	"			

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #1	PL# B13012.200	Rev. 0	Sheet 5 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R1	RCR05G153JS	RES,FXD,CMPSN,15K,1/8W,5%	AB		1
	R2	RCR05G202JS	" 2K 5%	AB		1
	R3	RCR05G103JS	" 10K 5%	AB		8
	R5	"	" 10K 5%	AB		
	R6	"	" 10K 5%	AB		
	R8	"	" 10K 5%	AB		
	R9	"	" 10K 5%	AB		
	R15	"	" 10K 5%	AB		
	R63	"	" 10K 5%	AB		
	R107	"	" 10K 5%	AB		
	R4	RCR05G472JS	" 4.7K 5%	AB		1
	R7	RCR05G203JS	" 20K 5%	AB		1
	R10	RCR05G332JS	" 3.3K 5%	AB		2
	R11	"	" 3.3K 5%	AB		
	R12	RNC50H4992FS	RES,FXD,PREC, 49.9K,1%,1/8W,REL=S	DALE	MIL-R-55182	3
	R13	"	"			
	R14	"	"			
	R20	RCR05G105JS	RES,FXD,CMPSN,1M,5%,1/8W	AB		3
	R21	"	" 1M, 1/8W	AB		
	R22	"	" 1M, 1/8W	AB		
	R24	RCR05G106JS	" 10M 1/8W	AB		1
	R25	RCR05G102JS	" 1K 1/8W	AB		11
	R29	"	" 1K 1/8W	AB		
	R31	"	" 1K 1/8W	AB		
	R32	"	" 1K 1/8W	AB		
	R36	"	" 1K 1/8W	AB		
	R38	"	" 1K 1/8W	AB		

APPLIED RESEARCH CORPORATION	Title	PL#	Rev.	Sheet
	Star Tracker PCB #1	B13012.200	0	6 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R39	RCR05G102JS	RES,FXD,CMPSN,1K,5%,1/8W	AB		
	R43	"	" 1K 1/8W	AB		
	R45	"	" 1K 1/8W	AB		
	R59	"	" 1K 1/8W	AB		
	R64	"	" 1K 1/8W	AB		
	R26	RCR05G220JS	" 22 Ohm, 1/8W	AB		6
	R27	"	" 22 Ohm, 1/8W	AB		
	R33	"	" 22 Ohm, 1/8W	AB		
	R34	"	" 22 Ohm,1/8W	AB		
	R40	"	" 22 Ohm,1/8W	AB		
	R41	"	" 22 Ohm, 1/8W	AB		
	R28	RCR05G2R7JS	" 2.7 Ohm,1/8W	AB		7
	R35	"	" 2.7 Ohm, 1/8W	AB		
	R42	"	" 2.7 Ohm, 1/8W	AB		
	R100	"	" 2.7 Ohm,1/8W	AB		
	R101	"	" 2.7 Ohm,1/8W	AB		
	R102	"	" 2.7 Ohm,1/8W	AB		
	R103	"	" 2.7 Ohm,1/8W	AB		
	R30	RCR05G226JS	" 22M, 1/8W	AB		6
	R37	"	" 22M, 1/8W	AB		
	R44	"	" 22M, 1/8W	AB		
	R65	"	" 22M, 1/8W	AB		
	R66	"	" 22M , 1/8W	AB		
	R67	"	" 22M , 1/8W	AB		
	R46	RCR05G4R7JS	" 4.7 Ohm , 1/8W	AB		3
	R47	"	" 4.7 Ohm , 1/8W	AB		
	R60	"	" 4.7 Ohm ,1/8W	AB		

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #1	PL# B13012.200	Rev. 0	Sheet 7 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R61	RCR07G4R7JS	RES,FXD,CMPSN,4.7 Ohm,5%,1/4W	AB		1
	R62	RCR05G753JS	" 75K, 5%,1/8W	AB		1
	R99	RCR05G563JS	" 56K 1/8W	AB		1
	R104	RCR05G684JS	" 680K 1/8W	AB		4
	R105	"	" 680K 1/8W	AB		
	R181	"	" 680K 1/8W	AB		
	R182	"	" 680K 1/8W	AB		
	R106	RCR05G101JS	" 100 Ohm 1/8W	AB		2
	R166	"	" 100 Ohm 1/8W	AB		
	R167	RCR05G100JS	10 Ohm 1/8W	AB		2
	R168	"	" 10 Ohm 1/8W	AB		

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #1	PL# B13012.200	Rev. 0	Sheet 8 of 8

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	U1	HA7-5147/883	OP AMPL,PREC,WIDEBAND	HAR		2
	U4	"	"	HAR		
	U2	DG303AAK/883	ANALOG SWITCH, CMOS, DUAL SPDT	SIX		2
	U8	"	"	SIX		
	U3	DG302AAK/883	ANALOG SWITCH, CMOS, DUAL DPST	SIX		1
	U5	OPA602SM -BI	BI OP AMPL, FET INPUT, WIDEBAND	BB		3
	U6	"	"	BB		
	U7	"	"	BB		
	U9	HI1-508/883	ANALOG MUX, CMOS, 8-CHNL	HAR		1
	U10	CD54HC00F3A	QUAD 2-INPUT NAND	HAR		1
		(= 8403701CA)				
	U11	14053B/BEAJC	TRIPLE 2-CHANNEL ANALOG MUX/DEMUX	MOT		3
	U18	"	"	MOT		
	U19	"	"	MOT		
	U12	CD4041UBF/3	QUAD TRUE-COMPLEMENT BUFFER	RCA		3
	U20	"	"	RCA		
	U21	"	"	RCA		
	U13	54AC14DMQB	HEX SCHMITT INVERTER	NAT		3
	U16	" (= 5962-8762401CA)	"	NAT		
	U17	"	"	NAT		
	U14	CD54HC14F3A	HEX SCHMITT INVERTER	HAR		1
		(= 8409101CA)				
	U15	CD54HC27F3A	TRIPLE 3-INPUT NOR	HAR		1
		(= 8404201CA)				

APPLIED RESEARCH CORPORATION	Title: Star Tracker PCB #2	PL#: B13013.200	Rev.: 0	Sheet 1 of 3

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C47	CCR05CH150JR	CAP,FXD,CER,TEMP-COMP,15pF,5%,200V,REL=R			1
	C48	M39014/01-1554	CAP,FXD,CER,0.1μF,20%,50V,REL=R			13
	C53	"	"			
	C54	"	"			
	C55	"	"			
	C56	"	"			
	C68	"	"			
	C69	"	"			
	C70	"	"			
	C71	"	"			
	C72	"	"			
	C73	"	"			
	C74	"	"			
	C75	"	"			
	C49	M39014/01-1300	CAP,FXD,CER,100pF,20%,200V,REL=R			1
	C76	M39014/01-1536	CAP,FXD,CER,0.01μF,20%,100V,REL=R			2
	C77	"	"			

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #2	PL# B13013.200	Rev. 0	Sheet 2 of 3

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R69	RCR05G102JS	RES,FXD,CMPSN,1K,1/8W,5%	AB		2
	R180	"	RES,FXD,CMPSN,1K,1/8W,5%	AB		
	R70	RCR05G103JS	RES,FXD,CMPSN,10K,1/8W,5%	AB		1
	R74	RCR05G106JS	RES,FXD,CMPSN,10M,1/8W,5%	AB		1
	R86	RCR05G104JS	RES,FXD,CMPSN,100K,1/8W,5%	AB		7
	R89	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R144	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R145	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R177	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R178	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R179	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R87	RCR05G753JS	RES,FXD,CMPSN,75K,1/8W,5%	AB		1
	R88	RCR05G153JS	RES,FXD,CMPSN,15K,1/8W,5%	AB		1
	R98	RCR05G105JS	RES,FXD,CMPSN,1M,1/8W,5%	AB		1
	R152	RCR05G471JS	RES,FXD,CMPSN,470 OHM,1/8W,5%	AB		3
	R153	"	RES,FXD,CMPSN,470 OHM, 1/8W,5%	AB		
	R163	"	RES,FXD,CMPSN, 470 OHM, 1/8W,5%	AB		
	R175	M8340102M2202GA	RESISTOR NETWORK,16PIN Dip	AB	MIL-R-83401	2
	R176	"	"	AB		

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #2	PL# B13013.200	Rev. 0	Sheet 3 of 3

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	U22	CD54HC14F3A	HEX SCHMITT INVERTER	HAR		1
	U23	LM124JB	QUAD OP AMPL	TI		2
	U30	"	"			
	U24	REF02AZ/883C	5 VOLT REFERENCE, 1mA	PMI		1
	U25	CD54HC08F3A	QUAD 2-INPUT AND	RCA		2
	U31	"	"	RCA		
	U26	CD54HC32F3A	QUAD 2-INPUT OR	RCA		1
	U27	LMC660AMJ/883C	QUAD OP AMPL, CMOS	NS		1
	U32	CD54HC74F3A	DUAL D FLIP-FLOP W SET & RESET			1
	U33	MAX543BMJA/883B	D-A CONVERTER, 12BIT SER. INPUT	MAX		1
	U34	CD4051BF3A	8-CHANNEL ANALOG MUX/DEMUX	HAR		1
	U35	AD7672TQ05/883	A-D CONVERTER, 12BIT	PMI		1
	U36	LM193H/883C	VOLTAGE COMPARATOR	NS		1
	U47	CD54HC00F3A	QUAD 2-INPUT NAND	HAR		1

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #3	PL# B13014.200	Rev. 0	Sheet 1 of 5

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C50	M39014/01-1554	CAP,FXD,CER,0.1μF,20%,50V,REL=R			6
	C51	"	"			
	C52	"	"			
	C61	"	"			
	C62	"	"			
	C63	"	"			
	C65	M39003/01-K5678J	CAP,FXD,ELCTLT,1μF,50V,5%,REL=S			1
	C66	M39014/01-1536	CAP,FXD,CER,0.01μF,20%,100V,REL=R			1
	C67	M39003/01-K2749J	CAP,FXD,ELCTLT,3.3μF,15V,20%,REL=R			1

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #3	PL# B13014.200	Rev. 0	Sheet 2 of 5

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	D24	JVIN5615	RECTIFIER,1A,200V	UNI		2
	D26	"	"			
	D25	JVIN4477	DIODE,ZENER,33V	UNI		1
	D28	JVIN4148-1	DIODE,SWITCHING			1
	D34	JXIN4624	DIODE,ZENER,4.7V	MSA		2
	D40	"	"	MSA		

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #3	PL# B13014.200	Rev. 0	Sheet 3 of 5

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	Q9	FRL9130D3	P MOSFET, 100V,0.55 OHM	HAR		2
	Q10	"	"			
	Q11	FRL130D3	N MOSFET, 100V, 0.18 OHM	HAR		1
	Q12	JV2N3019S	TRANSISTOR, NPN	MOT		1
	Q13	FRL9230D3	P MOSFET, 200V, 1.3 OHM	HAR		1
	Q14	JV2N2907A	TRANSISTOR, PNP	MOT		1
	U28	DG302AAK/883	ANALOG SWITCH, CMOS, DUAL DPST	SIX		1
	U29	CD54HC14F3A	HEX SCHMITT INVERTER	HAR		1
	U37	CD4051BF3A	8-CHANNEL ANALOG MUX/DEMUX	HAR		2
	U38	"	"			
	U41	MHF+2805S/883	DC-DC CONVERTER, 28 TO 5 VOLT	IP		1
	U42	MHF+2812D/883	DC-DC CONVERTER, 28 TO ± 12 VOLT	IP		1
	U43	LM11 7H/883C	VOLTAGE REGULATOR	NS		2
	U44	"	"			
	U45	46203	EMI FILTER, 750μH	PICO		1

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #3	PL# B13014.200	Rev. 0	Sheet 4 of 5

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R75	RCR05G104JS	RES,FXD,CMPSN,100K,1/8W,5%	AB		5
	R77	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R143	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R170	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R171	"	RES,FXD,CMPSN,100K,1/8W,5%	AB		
	R76	RCR05G683JS	RES,FXD,CMPSN,68K,1/8W,5%	AB		1
	R78	RCR05G331JS	RES,FXD,CMPSN,330 OHM, 1/8W,5%	AB		4
	R79	"	RES,FXD,CMPSN,330 OHM, 1/8W,5%	AB		
	R80	"	RES,FXD,CMPSN, 330 OHM,1/8W,5%	AB		
	R81	"	RES,FXD,CMPSN, 330 OHM,1/8W,5%	AB		
	R111	RCR05G105JS	RES,FXD,CMPSN,1M,1/8W,5%	AB		5
	R113	"	RES,FXD,CMPSN,1M,1/8W,5%	AB		
	R129	"	RES,FXD,CMPSN,1M,1/8W,5%	AB		
	R131	"	RES,FXD,CMPSN,1M,1/8W,5%	AB		
	R134	"	RES,FXD,CMPSN,1M,1/8W,5%	AB		
	R132	RCR05G684JS	RES,FXD,CMPSN,680K,1/8W,5%	AB		1
	R135	RCR05G302JS	RES,FXD,CMPSN,3K,1/8W,5%	AB		1
	R136	RCR05G152JS	RES,FXD,CMPSN,1.5K,1/8W,5%	AB		1
	R137	RCR05G6R2JS	RES,FXD,CMPSN,6.2 OHM	AB		1
	R139	RCR05G471JS	RES,FXD,CMPSN,470 OHM	AB		1
	R140	RCR05G474JS	RES,FXD,CMPSN,470K,1/8W,5%	AB		2
	R141	"	RES,FXD,CMPSN,470K,1/8W,5%	AB		
	R142	RCR05G473JS	RES,FXD,CMPSN,47K,1/8W,5%	AB		1
	R146	RCR05G123JS	RES,FXD,CMPSN,12K,1/8W,5%	AB		1
	R164	RCR05G221JS	RES,FXD,CMPSN,220 OHM	AB		3
	R165	"	RES,FXD,CMPSN,220 OHM	AB		

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #3	PL# B13014.200	Rev. 0	Sheet 5 of 5

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	R172	RCR05G221JS	RES,FXD,CMPSN,220 OHM,1/8W,5%	AB		
	R169	"	JUMPER WIRE			1
	R173	M8340102K4703GA*	See note below			1
	R174	M8340102K4703GB	RESISTOR NETWORK, 16 PIN DIP	AB	MIL-R-83401	1
		* Note: This resistor array to be replaced by 8 discrete resistors RCR05G474JS (470K 1/8W 5%)				

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #4	PL# B13015.200	Rev. 0	Sheet 1 of 2

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C1	M39014/01-1554	CAP,FXD,CER,0.1uF,20%,50V,REL=R			20
	C2	"	"			
	C3	"	"			
	C4	"	"			
	C5	"	"			
	C6	"	"			
	C7	"	"			
	C8	"	"			
	C9	"	"			
	C10	"	"			
	C11	"	"			
	C12	"	"			
	C13	"	"			
	C14	"	"			
	C15	"	"			
	C16	"	"			
	C17	"	"			
	C18	"	"			
	C19	"	"			
	C20	"	"			
	R1	RCR05G104JS	RES,FXD,CMPSN,100K,1/8W,5%	AB		5
	R2	"	"			
	R3	"	"			
	R4	"	"			
	R5	"	"			

APPLIED RESEARCH CORPORATION	Title	PL#	Rev.	Sheet
	Star Tracker PCB #4	B13015.200	0	2 of 2

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	U1	CD54HC283F3A	4-BIT BINARY ADDER	HAR		3
	U2	"	"	HAR		
	U3	"	"	HAR		
	U4	54HC374/BRAJC	OCTAL 3-STATE NON-INV. D FLIP-FLOP	MOT		6
	U5	" (=8407101RA)	"	MOT		
	U6	"	"	MOT		
	U7	"	"	MOT		
	U14	"	"	MOT		
	U19	"	"	MOT		
	U8	CD54HC00F3A	QUAD 2-INPUT NAND	HAR		1
	U9	(82C54)8406501JA	PROGRAMMABLE INTERVAL TIMER	HAR		1
	U11	CD54HC240F3A	OCTAL 3-STATE INV. BUFFER/LINE DR/LINE REC.	HAR		1
	U12	OW62256CD3-10	32Kx8 SRAM	OW		2
	U13	"	"	OW		
	U16	CD54HC273F3A(=8409901RA)	OCTAL D FLIP-FLOP	HAR		1
	U17	CD54HC245F3A	OCTAL 3-STATE NONINV. BUS TRANS.	HAR		2
	U18	" (=8408501RA)	"	HAR		
	U20	CD54HC74F3A	DUAL D FLIP-FLOP	HAR		2
	U21	" (=8405601CA)	"	HAR		
	U22	CD54HC244F3A(=8409601RA)	OCTAL 3-STATE NONINV. BUFFER	HAR		1
	U24	CD54HC138F3A(=8406201EA)	1-OF-8 DECODER	HAR		1

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #5	PL# B13016.200	Rev. 0	Sheet 1 of 2

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C1	M39014/01-1554	CAP,FXD,CER,0.1µF,20%,50V,REL=R			17
	C2	"	"			
	C3	"	"			
	C4	"	"			
	C5	"	"			
	C6	"	"			
	C7	"	"			
	C8	"	"			
	C9	"	"			
	C13	"	"			
	C14	"	"			
	C15	"	"			
	C16	"	"			
	C17	"	"			
	C18	"	"			
	C19	"	"			
	C20	"	"			
	C10	CCR05CG220JR	CAP,FXD,CER,TEMP-COMP,22pF,200V,5%			2
	C11	"	"			
		"	"			
	R3	RCR05G201JS	RES,FXD,CMPSN,200 OHM,1/8W,5%	AB		1
	R5		JUMPER WIRE			1
	R6	RCR05G104JS	RES,FXD,CMPSN,100K,1/8W,5%	AB		4
	R7	"	"			
	R10	"	"			
	R11	"	"			

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #5	PL# B13016.200	Rev. 0	Sheet 2 of 2

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	U9	MD80C86/B	16 BIT MICRO PROCESSOR	HAR		1
	U10	MD82C85/B	CLOCK CONTROLLER/GENERATOR	HAR		1
	U11	MD82C59A-5/B	INTERRUPT CONTROLLER	HAR		1
	U12	8406501JA(82C54)	PROGRAMMABLE INTERVAL TIMER	HAR		1
	U13	CD54HC573F3A	OCTAL 3-STATE NONINVERTING LATCH	HAR		5
	U14	" (=8512801RA)	"	HAR		
	U15	"	"	HAR		
	U16	"	"	HAR		
	U17	"	"	HAR		
	U18	CD54HC138F3A	1-OF-8 DECODER	HAR		3
	U19	" (=8406201EA)	"			
	U23	"	"			
	U20	CD54HC32F3A	QUAD 2-INPUT OR	HAR		2
	U21	" (=8404501CA)	"			
	U22	CD54HC08F3A	QUAD 2-INPUT AND	HAR		2
	U26	" (=8404701CA)	"			
	U24	CD54HC109F3A	DUAL J-$\overline{K}$ FLIP-FLOP	HAR		1
		(=8404501CA)				
	U25	CD54HC688F3A	8-BIT EQUALITY COMPARATOR	HAR		1
		(=5962-8681801RA)				
	U34	CD54HC00F3A	QUAD 2-INPUT NAND	HAR		1
		(=8403701CA)				
	Y1	CR64/U-12.000MHz	quartz crystal, 12MHz	US Crystal		1

APPLIED RESEARCH CORPORATION	Title Star Tracker PCB #6	PL# B13017.200	Rev. 0	Sheet 1 of 1

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C21	M39014/01-1594	CAP,FXD,CER,0.1uF,20%,50V,REL=S			8
	C22	"	"			
	C23	"	"			
	C24	"	"			
	C25	"	"			
	C26	"	"			
	C27	"	"			
	C28	"	"			
	R12	RCR05G201JS	RES,FXD,CMPSN,200 OHM,1/8W,5%	AB		3
	R13	"	"			
	R14	"	"			
		"	"			
	U1	OW62256CD3-10	32K x 8 SRAM "	OMNI-WAVE		4
	U2	"	"	"		
	U3	"	"	"		
	U4	"	"	"		
	U5	DM28C256-250/B	32K x 8 EEPROM	SEEQ		4
	U6	"	"	"		
	U7	"	"	"		
	U8	"	"	"		

APPLIED RESEARCH CORPORATION	Title: Star Tracker PCB #7	PL# B13018.200	Rev. 0	Sheet 1 of 1

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	C29	M39014/01-1594	CAP.FXD.CER.0.1µF.20%.50V.REL=S			10
	C30	"	"			
	C31	"	"			
	C32	"	"			
	C33	"	"			
	C34	"	"			
	C35	"	"			
	C36	"	"			
	C37	"	"			
	C38	"	"			
	D1	JV1N4153-1	DIODE			1
	R8	RCR05G473JS	RES,FXD,CMPSN,47K,1/8W,5%	AB		1
	R9	RCR05G392JS	RES,FXD,CMPSN,3.9K,1/8W,5%	AB		1
	U27	MD82C55A-5/B	PROGRAMMABLE PERIPHERAL INTERFACE	HAR		1
	U28	MA528151CB1	COMMUNICATION INTERFACE	MAR		1
	U29	DS26C31MJ/883C	QUAD DIFF LINE DRIVER	NS		1
	U30	DS26C32AMJ/883C	QUAD DIFF LINE RECEIVER	NS		1
	U31	CD54HC132F3A	QUAD 2-INPUT SCHMITT NAND	HAR		2
	U33	"	"			
	U32	14020 B/BEAJC883CGG	14-STATE BINARY RIPPLE COUNTER	MOT		1

APPLIED RESEARCH CORPORATION	Title Star Tracker, Electrical Parts, Mounted Off PCBs *	PL# B13019.200	Rev. 0	Sheet 1 of 1

Item Find	Ref. Des.	Part No.	Description/Notes	MFR	Spec.	Qty.
	U48	TC217-30	CCD array, 1134 x 486 elements, virtual	TI		1
			phase, with two analog memories			
	J1	G311P10-2P-B-15	CONNECTOR, D-TYPE, 15 PIN	Posi-tronic		1
	D18	LN261CAL(UR)	LED,RED, HI EFFIC	Pana-sonic		2
	D19	"	"	"		
	D20	LN54PA	LED,IR	"		1
	D21	HLMP-1790	LED,GREEN, HI EFFIC	Quality Technology		1
	RT1	311P18-10T-30R	THERMISTOR, + 0.5%, 30K	YSI		3
	RT2	"	"	"		
	RT3	"	"	"		
	U39	FC 0.45-32-05L	COOLER, PELTIER, WITH 30 GAUGE	MELCOR		2
			STRANDED WIRE			
	U40	"	"	"		
			* Note: These components belong electrically			
			to circuits on PCBs 1-3.			

Report Documentation Page

NASA National Aeronautics and Space Administration

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
A Low-Cost, CCD Solid State Star Tracker	December 23, 1992
	6. Performing Organization Code
7. Author(s) M. Chmielowski D. Wynne	8. Performing Organization Report No. R92-204
	10. Work Unit No.
9. Performing Organization Name and Address Applied Research Corporation 8201 Corporate Drive, Suite 1120 Landover, MD 20785	11. Contract or Grant No. NAS5-31169
	13. Type of Report and Period Covered Final Dec. 01/90-Nov. 30/92
12. Sponsoring Agency Name and Address NASA/Goddard Space Flight Center Greenbelt, MD 20771	14. Sponsoring Agency Code 745

15. Supplementary Notes

16. Abstract

ARC has developed an engineering model of a multi-star CCD-based tracker for space applications requiring radiation hardness, high reliability and low power consumption. The engineering unit compared favorably in functional performance tests to the standard NASA single-star tracker. Characteristics of the ARC star tracker are: field of view = 10° x 7.5°, sensitivity range of -1 to +5 star magnitude, NEA = 3" x 3", linearity = 5" x 5", and power consumption of 1-3 W (operating mode dependent). The software is upgradable through a remote link. The hardware-limited acquisition rate is 1-5 Hz for stars of +2 to +5 magnitude and 10-30 Hz for -1 to +2 magnitude stars. Mechanical and electrical interfaces are identical to the standard NASA star tracker.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Star Tracker, Attitude Sensors, Spacecraft Control	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified		

NASA FORM 1626 OCT 86