AL/CF-SR-1995-0002

ARMSTRONG LABORATORY

HUMAN SENSORY FEEDBACK LAB TESTBED: MBASSOCIATES EXOSKELETON/MERLIN ROBOT INTERFACE

Monty L. Crabill
Todd W. Mosher

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD
DAYTON OH 45440-3696

JUNE 1993

INTERIM REPORT FOR THE PERIOD MARCH 1990 TO JUNE 1992

Approved for public release; distribution is unlimited

AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

THIS SOFTWARE AND ANY ACCOMPANYING DOCUMENTATION IS RELEASED "AS IS." THE U.S. GOVERNMENT, ITS CONTRACTORS, AND THEIR SUBCONTRACTORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, CONCERNING THIS SOFTWARE AND ANY ACCOMPANYING DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL THE U.S. GOVERNMENT, ITS CONTRACTORS AND THEIR SUBCONTRACTORS BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR SOFTWARE AND ANY ACCOMPANYING DOCUMENTATION, EVEN IF INFORMED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGES.

Please do not request copies of this report from the Armstrong Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22314

DISCLAIMER

This Special Report is published as received and has not been edited by the Technical Editing staff of the Armstrong Laboratory.

TECHNICAL REVIEW AND APPROVAL

AL/CF-SR-1995-0002

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

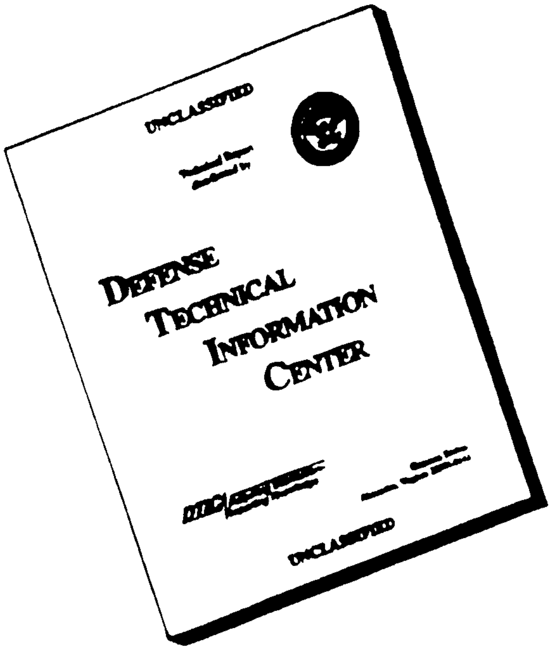
This technical report has been reviewed and is approved for publication.

FOR THE DIRECTOR

Thomas J Moore

THOMAS J. MOORE, Chief
Biodynamics and Biocommunications Division
Crew Systems Directorate
Armstrong Laboratory

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
June 1993

3. REPORT TYPE AND DATES COVERED
Interim - March 1990 to June 1992

4. TITLE AND SUBTITLE
Human Sensory Feedback Lab Testbed: MBAssociates
Exoskeleton/Merlin Robot Interface

5. FUNDING NUMBERS
C - F33615-89-C-0574
PE - 62202F
PR - 7231
TA - 38
WU - 08

6. AUTHOR(S)
Monty L. Crabill and Todd W. Mosher

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Systems Research Laboratories, Inc.
2800 Indian Ripple Road
Dayton OH 45440-3696

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Armstrong Laboratory, Crew Systems Directorate
Biodynamics and Biocommunications Division
Human Systems Center
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7901

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL/CF-SR-1995-0002

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the MBAssociates Exoskeleton/Merlin Robot Interface used in the Human Sensory Feedback Lab Testbed for studies of human performance in coarse positioning tasks. The dual-armed, seven-degree-of-freedom (DOF) exoskeleton, worn by an operator, serves as a master device to control the end-effector position/orientation of a six-DOF Merlin slave robot. Kinematic position transformation matrices are detailed for both master and slave devices, as is the approach used for operator control of a six-DOF robot using a seven-DOF exoskeleton. The use of a desktop 386-33MHz type personal computer as a central system controller, user interface, and software development platform is described.

14. SUBJECT TERMS
telepresence exoskeleton kinematics
telemanipulation Merlin Robot

15. NUMBER OF PAGES
190

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

TABLE OF CONTENTS

Page

1.0 INTRODUCTION 1

2.0 TECHNICAL DESCRIPTION 1

3.0 MBAssociates UNILATERAL EXOSKELETON 1
3.1 Backpack Computer Description 2
3.2 Backpack Software Description 3
3.3 Forward Kinematics for MBAssociates Exoskeleton . . . 4
3.3.1 Left Arm 5
3.3.2 Right Arm 11

4.0 INVERSE KINEMATICS FOR MERLIN ROBOT 17
4.1 Approach 17
4.2 High Speed Host Interface 19
4.3 Hardware and Configuration 19
4.4 Troubleshooting 20

5.0 HOST CONTROLLER COMPAQ 386 33MHz 21
5.1 Hardware and Configuration 21
5.2 MBA/Merlin Control Program 21

6.0 JR3 UNIVERSAL FORCE/MOMENT SENSOR SYSTEM 23

7.0 SUGGESTIONS FOR FUTURE WORK 23

8.0 REFERENCES 24

9.0 APPENDIX A: MBAssociates BACKPACK 25
9.1 Electronic Schematics 25
9.2 Timing Diagrams 39
9.3 Mechanical Drawings 51
9.4 Software Source Listing 69

10.0 APPENDIX B: MBA/MERLIN CONTROL SOFTWARE LISTING 85

11.0 APPENDIX C: Bit3 PC-AT ADAPTOR CARD JUMPER SETTINGS . . 182

12.0 APPENDIX D: SAFETY LIGHT FENCE SCHEMATICS 184

1.0 Introduction

Under contract F33615-89-C-0574 Task 08A "Human Sensory Feedback", Systems Research Laboratories provided engineering and scientific support to explore the utility of telerobotic systems in hazardous environments. A testbed was developed at the USAF Armstrong Laboratory's Human Sensory Feedback Lab at Wright Patterson AFB, to evaluate and research issues in course manipulation, bilateral teleoperation, force-control and man-machine interfaces.

2.0 Technical Description

The facility's testbed provides control of wrist position and end effector orientation for one six degree-of-freedom (DOF) American Cimflex Merlin industrial robot arm using a MBAssociates 7 DOF exoskeleton worn by a human operator. A Compaq 386 33 MHz computer serves as the host controller for the following functions:

- Computation of Cartesian position and orientation for various coordinate frames assigned at fixed locations on the exoskeleton, using joint angles received from the exoskeleton over a hardwire serial link at 38.4K bits/sec.

- Computation of joint angles required for the Merlin robot to achieve a desired wrist position and end effector orientation driven by the exoskeleton as a master.

- Provide user with system control functions.

3.0 MBAssociates Unilateral Exoskeleton

The MBAssociates exoskeleton was originally constructed by MBAssociates of San Ramon, CA under contract No. F08635-75-C-0027 with the United States Air Force, Armament Development and Test Center at Eglin AFB, Florida. The exoskeleton was part of a Manipulator Arms System comprised of an anthropomorphic two-armed teleoperator slave, powered by an electronically controlled hydraulic system and controlled by a two-armed exoskeletal type master. The master's grip and elbow joints on both arms provided force feedback by use of local hydraulic valves.

The Human Sensory Feedback Lab acquired the MBA exoskeleton through a long-term equipment loan arrrangement with the Department of Energy's Oak Ridge National Laboratories.

Several modifications were performed on the MBA exoskeleton to enhance its use as a standalone device in the Human Sensory Feedback Testbed.

Since initial use of the MBA exoskeleton was planned for position control of the Merlin robot, the exoskeleton's force feedback hydraulic actuators and manifold system were removed to reduce operator fatigue due to the effects of gravity. A microprocessor based computer, residing in the exoskeleton's backpack, was developed to read joint angles from either arm, using optical encoders located at each arm's 7 joints plus gripper control, and send them over a serial hardwire link to a host computer serving as the central controller for the exoskeleton and robot.

3.1 Backpack Computer Description

To provide the potential for the exoskeleton to be operated as a standalone device in a location remote from the Merlin robot, a 68000 microprocessor based computer with 2 asynchronous serial ports, battery backed SRAM and incremental optical encoder interface circuitry was developed to reside in the exoskeleton's backpack. Exoskeleton arm linkage joint angles, required to compute Cartesian position and orientation for exoskeleton coordinate frames, can be requested and received from a remote host controller over a 38.4K bits/sec hardwire serial link.

Joint angles are sensed by incremental optical encoders with 1024 counts/rev for 1:1 ratio joints and 120 counts/rev for gear/belt joints. Incremental optical encoder interface circuitry effectively increases the counts/rev to 4096 and 480 respectively, by counting the rising and falling edges of both quadrature channels from a single encoder. Incremental type encoders were selected over the original potentiometers to increase joint angle resolution and reduce suseptablity to electrical interference. Incremental type encoders were chosen over absolute types for cost considerations with the tradeoff being the need to initialize each joint of the exoskeleton after loss of power.

Serial communications between the MBAssociates exoskeleton and an external computer are specified electrically as RS-422 differential type drivers and receivers for longer distance and higher noise immunity reasons. The default bits/sec rate is 38.4K with 1 Meg bits/sec possible by selecting a higher oscillator base frequency for the DUART serial communications chip.

3.2 Backpack Software Description

The 68000 microprocessor executable code is downloaded from the host controller via the hardwire serial link to the MBA exoskeleton backpack computer. This provides development flexiblity in changing the software that initializes the incremental encoder interface circuitry, reads exoskeleton joint angle encoders and gripper switches on command, and communicates in real time with a host controller. Upon exoskeleton power-on/reset, a downloader routine, resident in EPROM (firmware) on the backpack's computer board, monitors the RS-422 serial port for a download command from the host computer and handles the transfer of 68000 executable code to an area in battery-backed SRAM. Once the download process is complete, the on-board 68000 CPU exits the downloader routine and begins execution of the downloaded code in SRAM.

The downloader software was developed on a Hewlett-Packard 64000 Development System with a 68000 microprocessor emulation pod. The emulation pod plugs directly into the 68000 microprocessor socket on the backpack computer board to allow development and testing of code on the target hardware, with finalized downloader code programmed into EPROM. All downloader software is archived on a Hewlett-Packard 9134 hard disk drive associated with the HP 64000 Development System.

Development of a backpack computer program to be downloaded is initiated using Aztec_C high level "C" programming language with Borland Turbo C II as a development tool on a Compaq 386 PC. Using the Aztec_C cross-compiler and linker, 68000 executable code is produced for downloading to the MBA exoskeleton backpack computer.

The downloading of 68000 executable code from the host Compaq 386 PC to the MBA exoskeleton backpack computer is handled by routines called by the program **merlin.exe** on the Compaq 386.

3.3 Forward Kinematics for MBAssociates exoskeleton

Forward kinematic transformation matrices were developed using the Denavit-Hartenberg convention as described in Reference[1].

General form of ${}^{i-1}{}_{i}T$:

$$
{}^{i-1}{}_{i}T = \begin{vmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

Where:

a_i = the distance from Z_i to Z_{i+1} measured along X_i

α_i = the angle between Z_i and Z_{i+1} measured along X_i

d_i = the distance from X_{i-1} to X_i measured along Z_i

θ_i = the angle between X_{i-1} and X_i measured about Z_i

c = cosine

s = sine

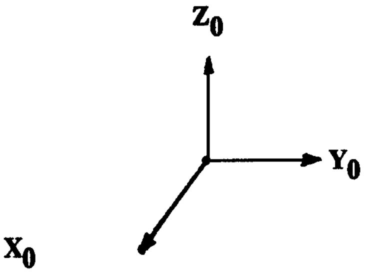
Location of Frame {i}'s Z_i axis was chosen to be along the optical encoder output shaft of joint {i} The direction of X_i was chosen to insure that the value of θ_i increases/decreases in the same direction as the encoder counts. Encoder counts increase with clockwise rotation of the shaft as viewed from the encoder body out towards the shaft.

Frame assignments and matrices are shown for each joint of each arm to notate the sparse composition of most matrices and to relation program execution times for kinematic computations.

3.3.1 Left Arm

Reference Frame {0}

The Origin of Reference Frame {0} is at the countersunk screw located top center on the MBA's backplate. (See figure 1)



$a_0 = 0$

$\alpha_0 = 0°$

d_0 = Does not exist

Θ_0 = Does not exist

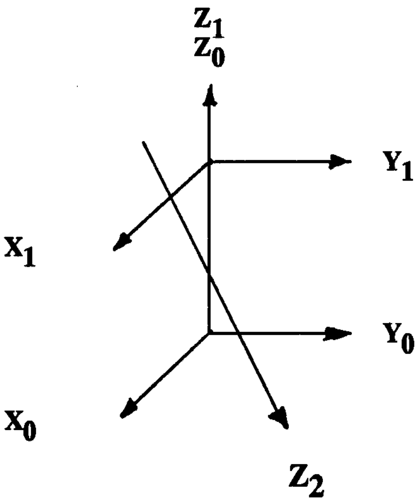
Left Frame {1}

$a_1 = +l_3$

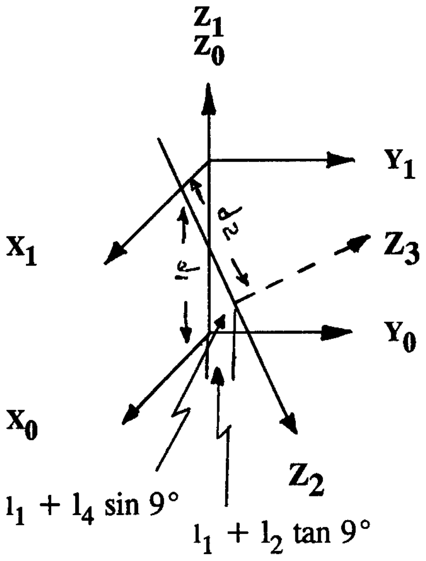
$\alpha_1 = +189°$

$d_1 = (l_1 + l_2 \tan 9°)/\tan 9°$

$\Theta_1 = 0°$



YZ View



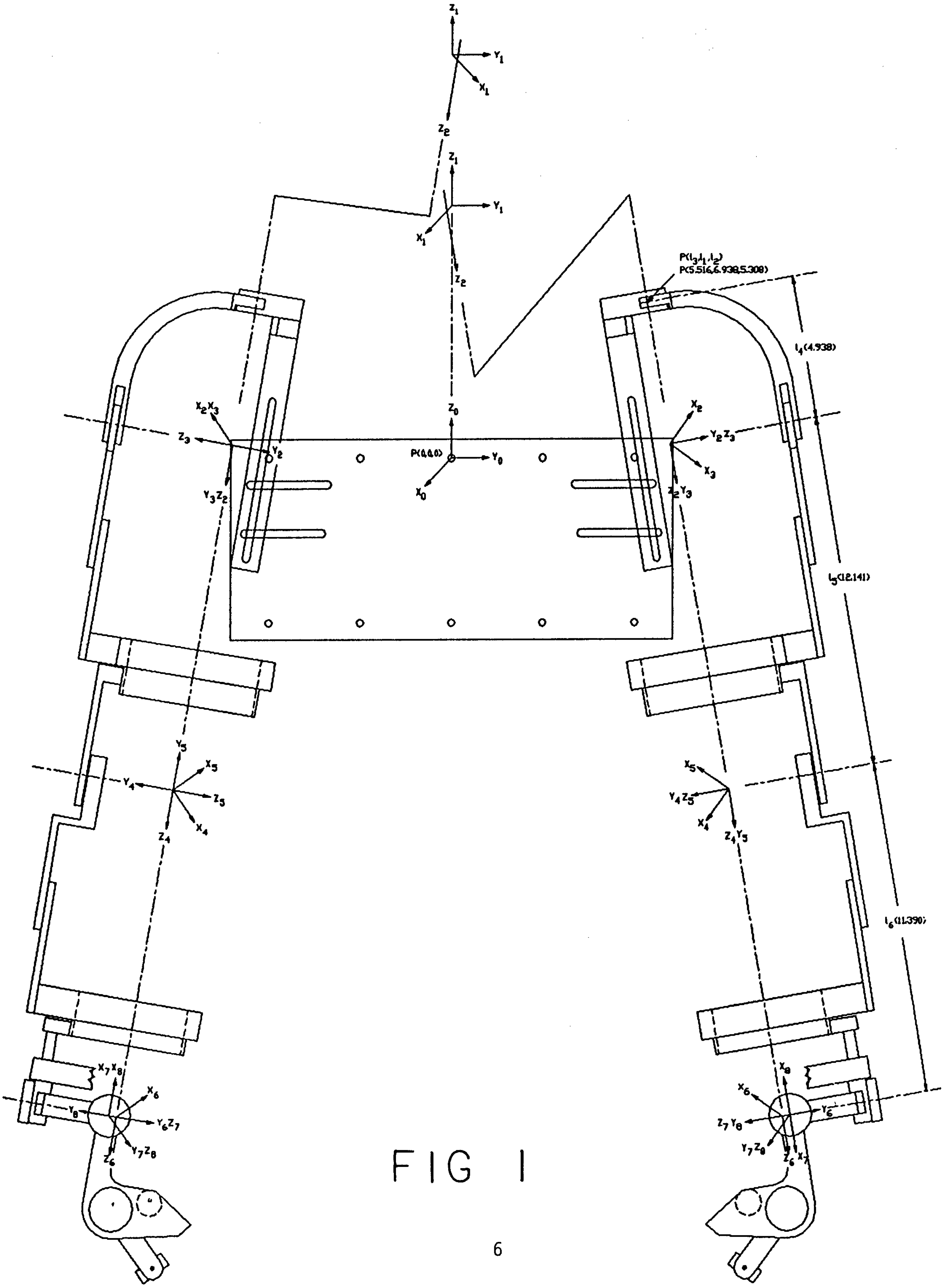

Z1
Y1
X1
Z2
Z1
Y1
X1
Z2
P(l3,l1,l2)
P(5.516,6.938,5.308)
l4(4.938)
Z0
X2X3
Z3
Y2
Y3Z2
P(0,0,0)
Y0
X0
X2
Y2Z3
X3
Z2Y3
l5(12.141)
Y5
X5
Y4
Z5
X4
Z4
X5
Y4Z5
X4
Z4Y5
l6(11.390)
X7X8
X6
Y8
Y6Z7
Y7Z8
Z6
X8
X6
Y6
Z7Y8
Y7Z8
Z6
X7

FIG 1

To solve for d_1,

$d_1 = (l_1 + l_2 \tan 9°)/ \tan 9°$

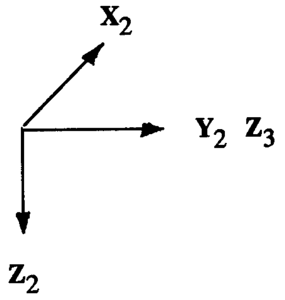
where : (See Figure 1)

$l_1 = 6.9375$ distance from Z_0 in $+Y_0$ direction
$l_2 = 5.3075$ distance from Z_0 in $+Z_0$ direction
$l_3 = 5.5156$ distance from back plate to Z_2 in $+X_0$ direction
$l_4 = 4.9375$
$l_5 = 12.141$

$$ {}^{0}_{1}\mathbf{T} = \left|\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & (l_1 + l_2 \tan 9°) / \tan 9° \\ 0 & 0 & 0 & 1 \end{matrix}\right| $$

Left Frame {2}

Located at Shoulder Azimuth and Shoulder Elevation Axes point of intersection.



$a_2 = 0$

$\alpha_2 = -90°$

$d_2 = (l_1 + l_4 \sin 9°)/\sin 9°$

Θ_2 = Shoulder Azimuth Joint

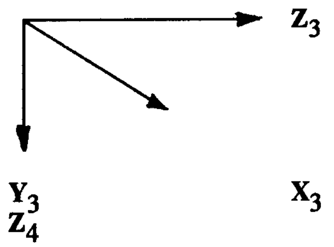
+90° all the way back

+225.9° all the way forward

$$ {}^{1}_{2}\mathbf{T} = \left|\begin{matrix} c\Theta_2 & -s\Theta_2 & 0 & l_3 \\ -s\Theta_2(.988) & -c\Theta_2(.988) & .156 & .156(l_1+l_4\sin9°)/\sin9° \\ -s\Theta_2(.156) & -c\Theta_2(.156) & -.988 & -.988(l_1+l_4\sin9°)/\sin9° \\ 0 & 0 & 0 & 1 \end{matrix}\right| $$

Left Frame {3}

Located at the shoulder Elevation & Upper Arm Roll Intersect.



$a_3 = 0$

$\alpha_3 = -90°$

$d_3 = -.0938$

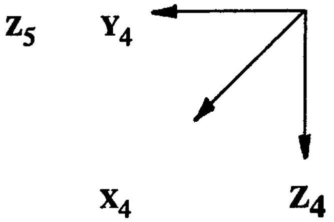
Θ_3 = Shoulder Elevation

+178° all the way back
+46.4° all the way forward

$$ {}^{2}_{3}\mathbf{T} = \left| \begin{array}{cccc} c\Theta_3 & -s\Theta_3 & 0 & 0 \\ 0 & 0 & 1 & -.0938 \\ -s\Theta_3 & -c\Theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right| $$

Left Frame {4}

Located at upper arm roll and elbow intersect.



$a_4 = 0$

$\alpha_4 = -90°$

$d_4 = l_5$

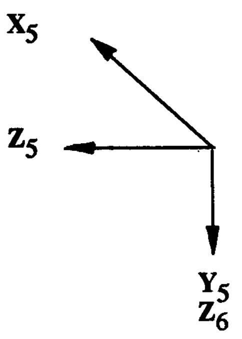
Θ_4 = Upper arm Roll

-43.9° CW Hard Stop
173.7° CCW Hard Stop

$$ {}^{3}_{4}\mathbf{T} = \left| \begin{array}{cccc} c\Theta_4 & -s\Theta_4 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ -s\Theta4 & -c\Theta4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right| $$

Left Frame {5})

Located at Elbow and Lower Arm Roll Intersect



$a_5 = 0$

$\alpha_5 = -90°$

$d_5 = -.313$

Θ_5 = Elbow Joint

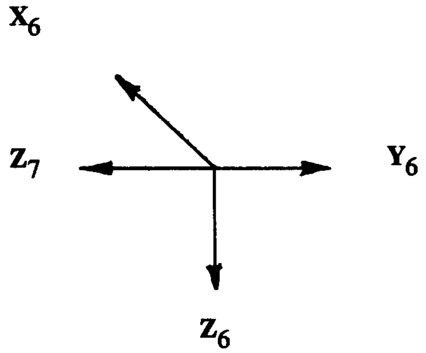
+46° All the way back

+316.1° All the way up

$$ {}^{4}_{5}T = \begin{vmatrix} c\Theta_5 & -s\Theta_5 & 0 & 0 \\ 0 & 0 & 1 & -.313 \\ -s\Theta_5 & -c\Theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Left Frame {6}

Located at Lower Arm Roll and Wrist Radial Intersect



$a_6 = 0$

$\alpha_6 = 90°$

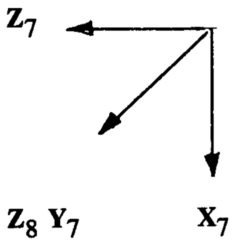
$d_6 = l_6 = 11.250$

Θ_6 = Lower Arm Roll Joint

$$ {}^{5}_{6}T = \begin{vmatrix} c\Theta_6 & -s\Theta_6 & 0 & 0 \\ 0 & 0 & 1 & l_6 \\ -s\Theta_6 & -c\Theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Left Frame {7}

Located at Wrist Radial and Wrist Flex Intersect



$a_7 = 0$

$\alpha_7 = -90°$

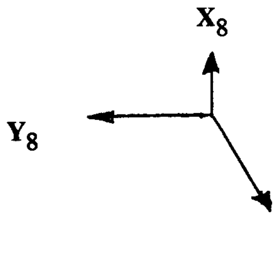
$d_7 = -.184$

$\Theta_7 =$ Wrist Radial

$$ {}^{6}_{7}\mathbf{T} = \left| \begin{matrix} c\Theta_7 & -s\Theta_7 & 0 & 0 \\ 0 & 0 & -1 & .184 \\ s\Theta_7 & c\Theta_7 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \right| $$

Left Frame {8}

Located at Wrist Flex



$a_8 =$ Not Defined

$\alpha_8 =$ Not Defined

$d_8 = 0$

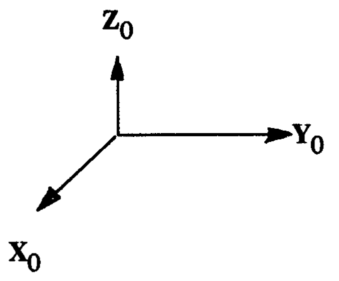
$\Theta_8 =$ Wrist Flex

$$ {}^{7}_{8}\mathbf{T} = \left| \begin{matrix} c\Theta_8 & -s\Theta_8 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\Theta_8 & -c\Theta_8 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \right| $$

3.3.2 Right Arm

Reference Frame {0}

The Origin of Reference Frame {0} is at the countersunk screw located top center on the MBA's backplate. (See figure 1)



$a_0 = 0$

$\alpha_0 = 0°$

d_0 = Does not exist

Θ_0 = Does not Exist

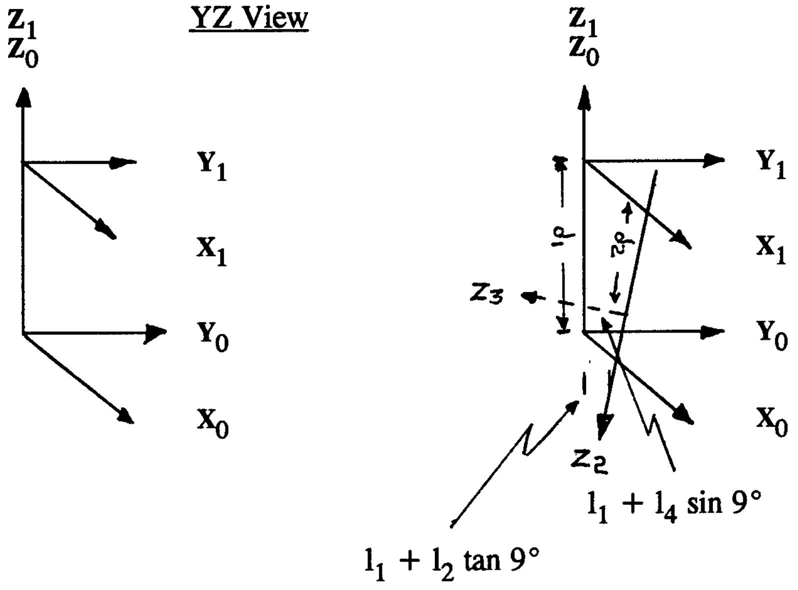
Right Frame {1}

$a_1 = +l_3$

$\alpha_1 = +171°$

$d_1 = (l_1 + l_2 \tan 9°)/\tan 9°$

$\Theta_1 = 0°$



To solve for d_1,

$$d_1 = (l_1 + l_2 \tan 9°)/ \tan 9°$$

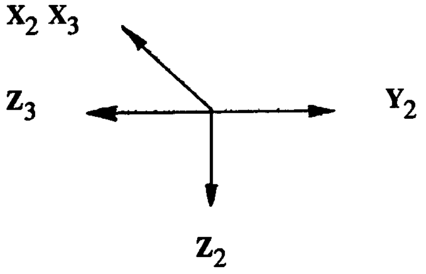
where:

l_1 = 6.9375 distance from Z_0 in $-Y_0$ direction
l_2 = 5.3075 distance in $-Z_0$ direction from Z_0
l_3 = 5.5156 distance in X_0 direction from back plate to Z_2
l_4 = 4.9375
l_5 = 12.141

$$^{0}_{1}\mathbf{T} = \left| \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & (l_1 + l_2 \tan 9°) / \tan 9° \\ 0 & 0 & 0 & 1 \end{array} \right|$$

Right Frame {2}

Located at Shoulder Azimuth and Shoulder Elevation Axes point of intersection.



$a_2 = 0$

$\alpha_2 = 90°$

$d_2 = (l_1 + l_4 \sin 9°)/\sin 9°$

Θ_2 = Shoulder Azimuth Joint

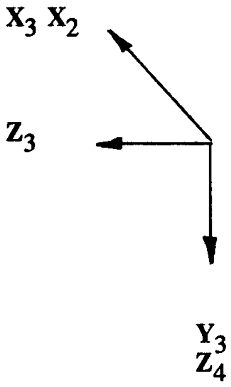
+270° all the way back

+134.1° all the way forward

$$^{1}_{2}\mathbf{T} = \left| \begin{array}{cccc} c\Theta_2 & -s\Theta_2 & 0 & +l_3 \\ -s\Theta_2(.988) & -c\Theta_2(.988) & .156 & .156(l_1 + l_4 \sin 9°)/\sin 9° \\ -s\Theta_2(.156) & -c\Theta_2(.156) & -.988 & -.988(l_1 + l_4 \sin 9°)/\sin 9° \\ 0 & 0 & 0 & 1 \end{array} \right|$$

Right Frame {3}

Located at the shoulder Elevation & Upper Arm Roll Intersect.



$a_3 = 0$

$\alpha_3 = -90°$

$d_3 = -.032$

Θ_3 = Shoulder Elevation

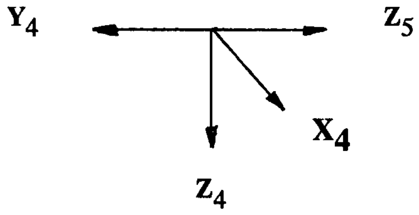
+2° back

+133.6° forward

$$ {}^{2}_{3}T = \left| \begin{array}{cccc} c\Theta_3 & -s\Theta_3 & 0 & 0 \\ 0 & 0 & -1 & -.032 \\ -s\Theta_3 & c\Theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right| $$

Right Arm Frame {4}

Located at upper arm roll and elbow intersect.



$a_4 = 0$

$\alpha_4 = 90°$

$d_4 = l_5$

Θ_4 = Upper arm Roll

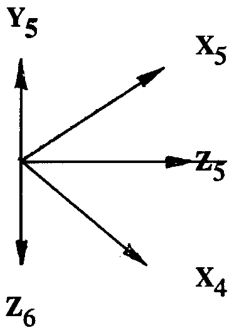
0° Rotated Inward

217.6° Rotated Outward

$$ {}^{3}_{4}T = \begin{vmatrix} c\Theta_4 & -s\Theta_4 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ -s\Theta_4 & -c\Theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Right Frame {5}

Located at Elbow and Lower Arm Roll Intersect



$a_5 = 0$

$\alpha_5 = 90°$

$d\,5 = 0$

Θ_5 = Elbow Joint

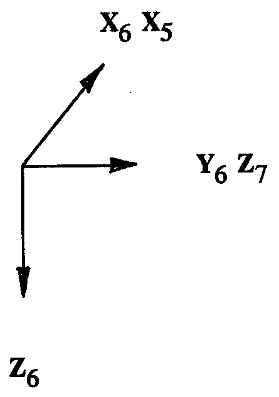
+46° All the way up

+316.1° All the way back

$$ {}^{4}_{5}T = \begin{vmatrix} c\Theta_5 & -s\Theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\Theta_5 & c\Theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Right Arm Frame {6}

Located at Lower Arm Roll and Wrist Radial Intersect



$a_6 = 0$

$\alpha_6 = -90°$

$d_6 = l_6 = 11.330$

Θ_6 = Lower Arm Roll Joint

Range of Motion:

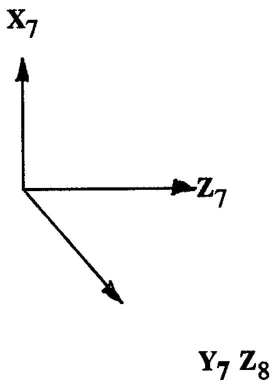
-97.32° Inward

119.91° Outward

$$ {}^5_6T = \begin{vmatrix} c\Theta_6 & -s\Theta_6 & 0 & 0 \\ 0 & 0 & -1 & -l_6 \\ s\Theta_6 & c\Theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

Frame {7}

Located at Wrist Radial and Wrist Flex Intersect



$a_7 = 0$

$\alpha_7 = -90°$

$d_7 = -.160$

Θ_7 = Wrist Radial

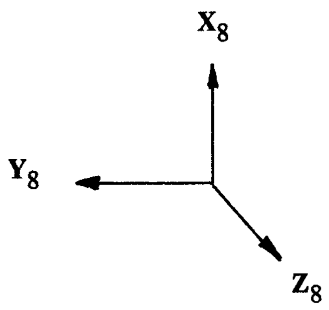
73.84° Up

105.57° Down

$$
{}^{6}_{7}\mathbf{T} = \begin{vmatrix} c\Theta_7 & -s\Theta_7 & 0 & 0 \\ 0 & 0 & 1 & -.160 \\ -s\Theta_7 & -c\Theta_7 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

Frame {8}

Located at Wrist Flex



a_8 = Not Defined

α_8 = Not Defined

$d_8 = 0$

Θ_8 = Wrist Flex

-47.35° Outward

42.74° Inward

$$
{}^{7}_{8}\mathbf{T} = \begin{vmatrix} c\Theta_8 & -s\Theta_8 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\Theta_8 & -c\Theta_8 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

4.0 Inverse Kinematics for Merlin Robot

The position of Merlin's wrist (coincident origins of frames {4}{5}{6}) with respect to Merlin's fixed reference frame {0} is driven by the position of MBA frame {6} origin (wrist radial) with respect to MBA's fixed reference frame {0}.

Orientation of Merlin's wrist (tool roll frame {6}) with respect to Merlin's fixed reference frame {0} driven by the orientation of the MBA exoskeleton's wrist (wrist flex frame {8} with respect to the MBA's exoskeleton's fixed reference frame {0}.

4.1 Approach

Merlin's wrist position determined by driving Merlin's frames {1} (waist), {2} (shoulder), and {3} (elbow) directly from the results of the following inverse kinematics.

Utilize Z-Y-Z Euler angle convention to describe Merlin's wrist orientations. Z-Y-Z Euler angles with respect to Merlin's elbow frame {3} can be used directly to drive Merlin's wrist roll, wrist flex & tool roll revolute joints Θ_4, Θ_5, Θ_6. respectively. Derivation of Z-Y-Z Euler angles to describe the MBA's wrist (wrist flex frame {8}) starting with MBA {8} coincident with MBA elbow frame {5} will result in an orientation of the Merlin wrist {6} different from the MBA wrist {8}. Therefore, a phantom frame {A} was defined on the MBA with an origin coincident at MBA {6}. Now the orientation of MBA frame {8} wrist flex can be described by Z-Y-Z Euler angles from frame {A}, thereby yielding results that can be used directly to drive Merlin's $\Theta_{4,}$ $\Theta_{5,}$ Θ_6 to orient Merlin's wrist.

MBA Exoskeleton	Merlin Robot
{5} = elbow {A} = {8} = wrist flex	{3} = elbow

Want :	$\text{Global}_{0}R$	=	$\text{Global}_{0}R$
Know :	${}^{0}_{5}R$		${}^{0}_{3}R$
Need :	${}^{0}_{A}R$	=	${}^{0}_{3}R$
Solve for:			
	${}^{0}_{A}R = {}^{0}_{5}R\ {}^{5}_{A}R$	=	${}^{0}_{3}R$
	${}^{5}_{A}R$	=	${}^{0}_{5}R^{-1}\ {}^{0}_{3}R$
Need :	${}^{A}_{8}R$		
Know :	${}^{5}_{8}R$ & ${}^{5}_{A}R$		
So :			
	${}^{A}_{8}R = {}^{A}_{5}R\ {}^{5}_{8}R$		
Solve for:			
	${}^{A}_{5}R = {}^{5}_{A}R^{-1}$		
Then:			

$$
{}^{A}_{8}R = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix} = R_{z\,y\,z}\ (\alpha,\beta,\tau)
$$

$\beta = \text{Atan2}\ (\sqrt{}\ (r_{31}{}^{2} + r\ {}_{32}{}^{2})\ ,\ r_{33}) =$ wrist flex (Θ_4)

$\alpha = \text{Atan2}\ (\ r_{23}\ /\ \sin\beta\ ,\ r_{13}\ /\ \sin\beta\ ) =$ wrist roll (Θ_5)

$\tau = \text{Atan2}\ (\ r_{32}\ /\ \sin\beta\ ,\ -r_{31}\ /\ \sin\beta\ ) =$ tool roll (Θ_6)

4.2 High Speed Host Interface

The High Speed Host Interface developed by American Cimflex provides a parallel shared memory interface between the Compaq 386 host controller computer and the Merlin controller's master CPU. The HSHI software, executing on the Merlin controller's master CPU board, performs motion related commands and provides status for each robot joint.

4.3 Hardware and Configuration

The American Cimflex MR6500 Merlin robot is equipped with stepper type motors for each of the six joints. The Waist, Shoulder, and Elbow joints are actuated with Sigma 802-D42-802-8780 steppers, and the three wrist joints are actuated by Sigma 802-D28698 steppers. Sigma Motion Control Products was bought out by Pacific Scientific in Rockford, IL 815-226-3100.

The Bit3 PC/AT to VME link is configured with Dual Port Ram (DPR) loacated at address 100000H on VME board. Jumper diagrams apprear in Appendix C. Note: PC Memory extension software drivers such as EMM386 may not be compatible with PC side of Bit3. (See Reference [6] Bit3 IBM PC/AT VME Adaptor Model 403 Manual)

HAL Engineering VME to Versabus Adapter Card installed in Merlin Control Cabinet Versabus. Bit3 VME card is installed into Adapter card.

HSHI Eproms (2716 2Kx8) must be installed for each axis on Merlin's Axis Control Boards. (See Reference [4] HSHI Manual)

Editing a HSHI Executive System Disk

(See Reference [4] ARBASIC Manual)

1) MAGIX SYSTEMS DISK in Merlin Controller's DRIVE 0
2) AR-BASIC in DRIVE 1
3) Front Panel key switch in "PROG" position
4) Power on Merlin Controller Circuit Breaker
5) Depress Front Panel "Power On" button
6) MAGIX OS and AR-BASIC will auto-load
7) If Front Panel "CAL" button blinks, Merlin's calibration procedure must be performed by engaging the motors and pressing the Front Panel "CAL" button. Merlin will jostle all joints and return AR-BASIC prompt " < " on the CRT.
8) With the AR-BASIC prompt " < " on the CRT, replace the AR-BASIC disk with the HSHI disk to be edited.

9) To tell MAGIX OS to track disk changes, type "F_LIST /etc/"

10) Type "LOAD /usr/etc/sysexec.def". The HSHI Executive System file contents will appear on the CRT's top window.

11) To edit the file for the HSHI Menu Option:
- Press "F1" to enter CRT edit window
- Edit the file as follows:

```
!
! boot HSHI initially ...
! Echo Booting HSHI
!   hshi -m 100000
!   Kill JOAN
! Host mode
```

12) To edit the file for auto-boot of HSHI Host mode @ power-on

```
!
! boot HSHI initially ...
! Echo Booting HSHI
      hshi -m 100000
      Kill JOAN
! Host mode
```

13) Press "F1" to return to AR-BASIC's command line on CRT

14) Type "SAVE /usr/etc/sysexec.def"

15) Type "M_DELETE $pgm" to remove file from editor memory

16) Type "BYE" to exit AR-BASIC

4.4 Troubleshooting

Merlin Robot

Sympton - Joint drifts/ non responsive

Correction - If green LED on motor controller card is not "on", replace/repair motor controller card.

- If green LED on motor controller card is "on", remate all motor and encoder connectors in junction box at robot base.

Intecolor 2400 Series Terminal

Refer to "Maintenance Manual for Intecolor 2400 Series Terminals with ColorTrend 220 Supplement"

5.0 Host Controller Compaq 386 33MHz

5.1 Hardware and Configuration

(See Figure 2)

A Compaq 386 33 MHz desktop PC serves as the system host controller, software development computer and operator interface to the testbed.

The MBA exoskeleton is linked to the system host via a 38.4 K bits/sec RS-422 asynchronous serial link.

The Merlin robot controller is linked to the system host through a parallel link using dual port ram provided by a Bit 3 Model 403 PC/AT-VME Adaptor.

The JR3 Univeral Force/Moment System is linked to the system host controller via a 9600 bits/sec RS-232 asynchronous serial link.

5.2 MBA/Merlin control program

Program execution times for **merlin.exe** on a Compaq 386 33 Mhz computer with floating point numeric coprocessor:

-	MBA exoskeleton's forward kinematic algorithm	0.732 mS
-	Merlin robot inverse kinematic algorithm	0.960 mS
-	Serial transmission time to transfer 7 joint angle readings from exoskeleton right arm. This time is transparent to total control loop time as it is interrupt driven.	
	7 joints * 2 integer bytes/joint * 10 bits/integer value transmitted * 38.4 Kbits/sec	(3.64 mS)
-	Time for serial interrupt servicing, pre-processing of joint angle readings for forward kinematics, limit and error checking on inverse kinematic results and operator display updates.	8.18 mS
	Total control loop time	9.87 mS

The 9.87 mS (101 Hz) total control loop time is greater than the Merlin's 4 mS (250 Hz) servo loop time achievable under optimal HSHI operation. This difference in loop times means the Merlin is not receiving motor encoder servo values fast

MBA/MERLIN TESTBED BLOCK DIAGRAM

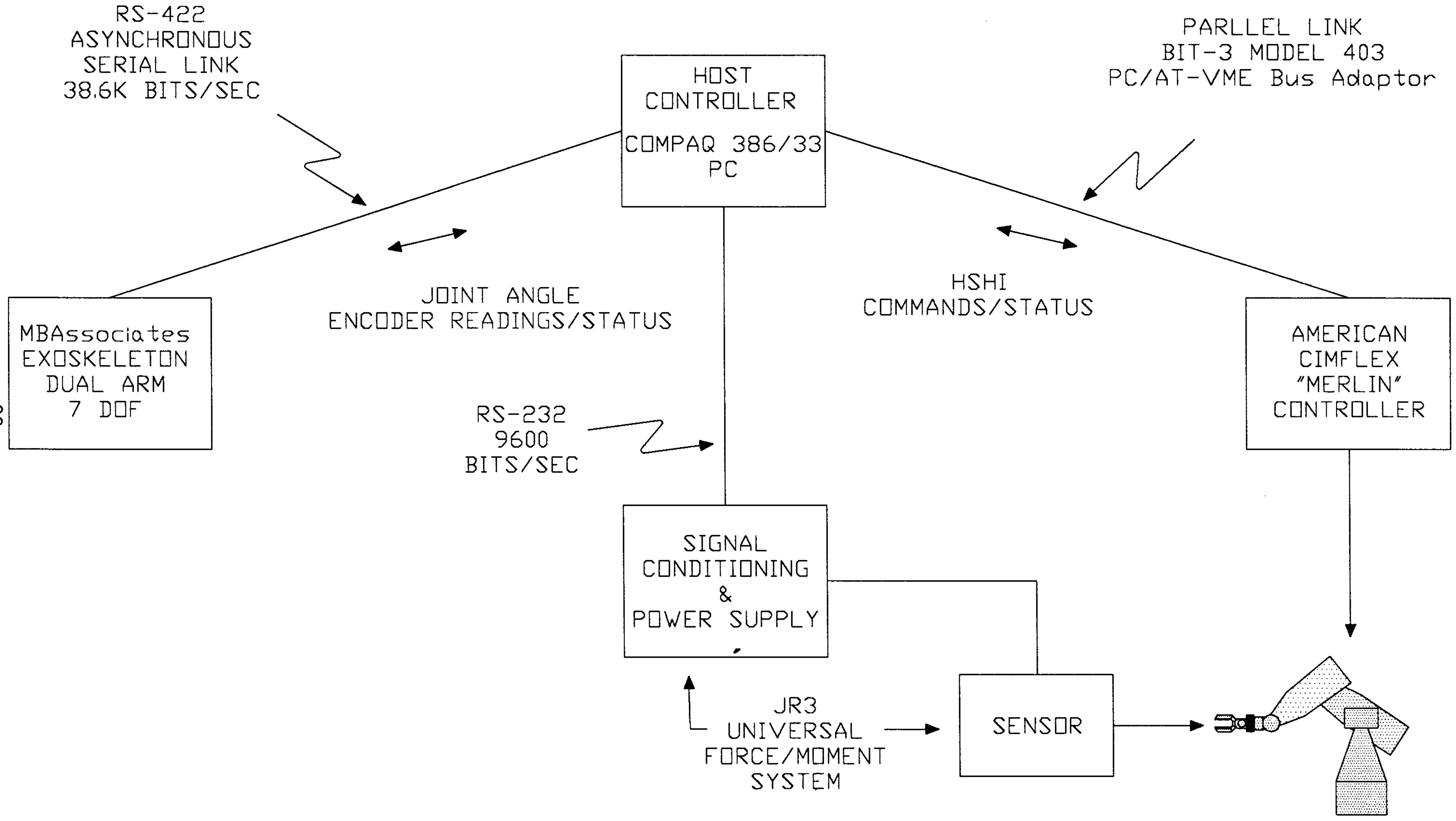


Figure 2

enough from the host controller. Better real time response in positioning motions may result by decreasing this difference to slightly less that 8ms or 4ms to hit the Merlin's 4ms update window.

6.0 JR3 Universal Force/Moment Sensor System

A six axis force/moment sensor system mounted on the Merlin's tool faceplate, provides 3 force components (F_x,F_y,F_z) and 3 moment components (M_x,M_y,M_z) from the Merlin's end effector interaction with it's environment. Additionally, the sensor is used as a safety limit device to avoid damage to the Merlin's gripper and/or a testbed fixture. Force/Moment data is transmitted to the host system controller via a 9600 bits/sec RS-232 serial link.

7.0 Suggestions for Future Work

Currently, course positioning experiments using the MBA exoskeleton to control the Merlin robot are somewhat limited by the response of the Merlin to rapid changes in position. With the Merlin end effector's position and orientation controlled by individual joint axis controllers servoing only to achieve a commanded motor encoder count (motor shaft position) using a fixed velocity profile, Merlin's end effector response is degraded by velocity lag or positional overshoot.

The HSHI option provides a Motor Velocity Command that could be utilized to improve Merlin's response. This would require the host system controller (Compaq 386) to close the control loop on the robot side by commanding the Merlin's individual joint axis controllers to achieve wrist position/velocity and end effector orientation/angular velocity. Merlin motor shaft position would be used in the servo loop.

8.0 References

[1] Craig, John J. Introduction to Robotics: Mechanics and Control Reading, Massachusetts: Addison-Wesley Publishing Co., 1986

[2] D'Souza, A. Frank/ Vijay K. Garg Advanced Dynamics Modeling and Analysis Englewood Cliffs, New Jersey: Prentice-Hall Inc. 1984

[3] "Operations and Maintenance Manual Manipulator Arms System," MBA MBAssociates, San Ramon, CA, 6 April 1976

[4] American Cimflex Operating/ARBASIC/Service/HSHI Manuals

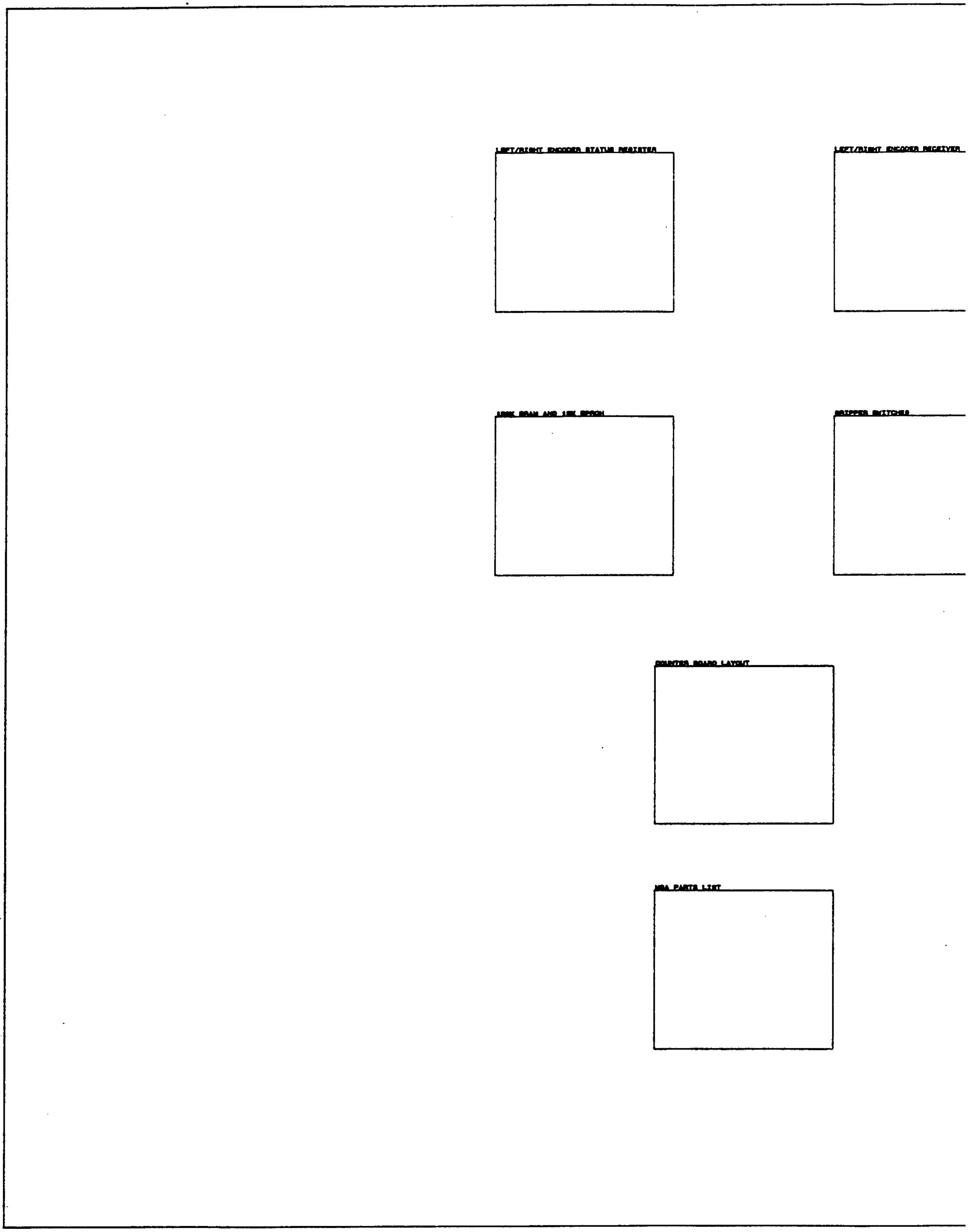
[5] JR3 Universal Force/Moment Sensor System Manual

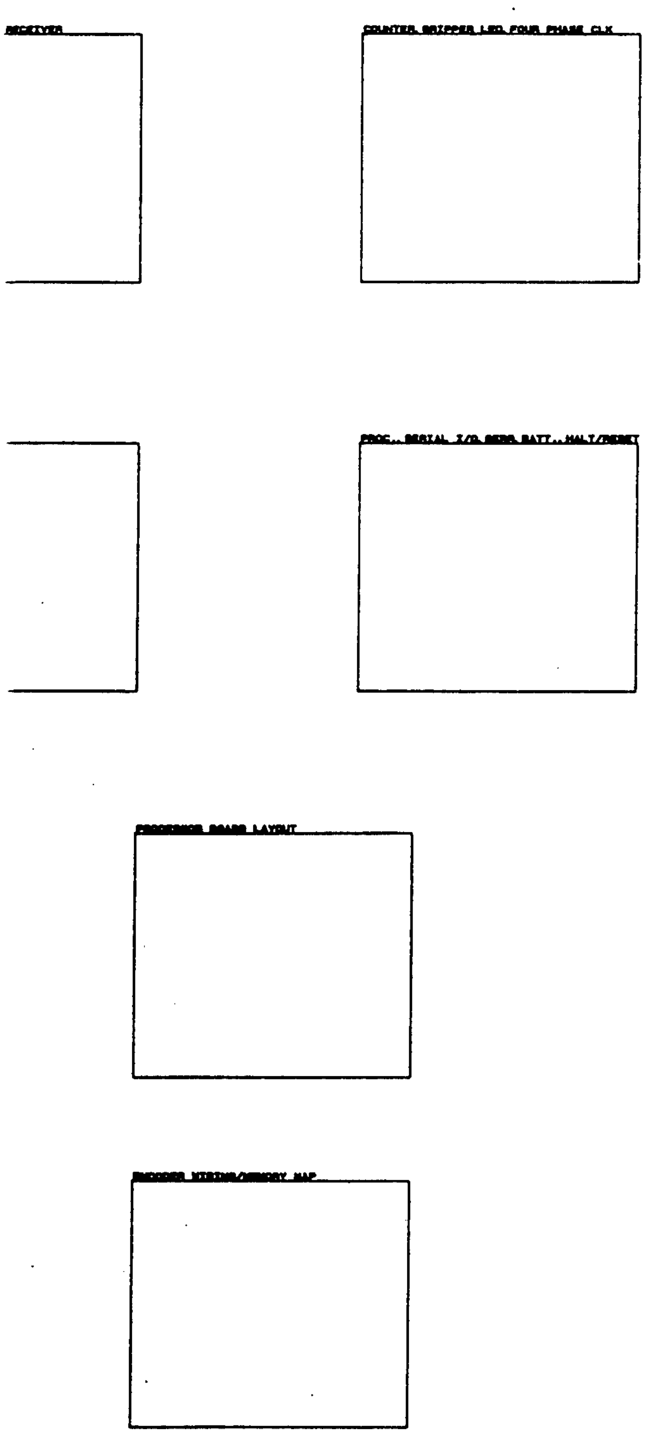
[6] Bit3 Corporation IBM PC/AT VME Adaptor Model 403 Manual.

9.0 Appendix A: MBAssociates Backpack

9.1 Electronic Schematics

This page intentionally left blank.

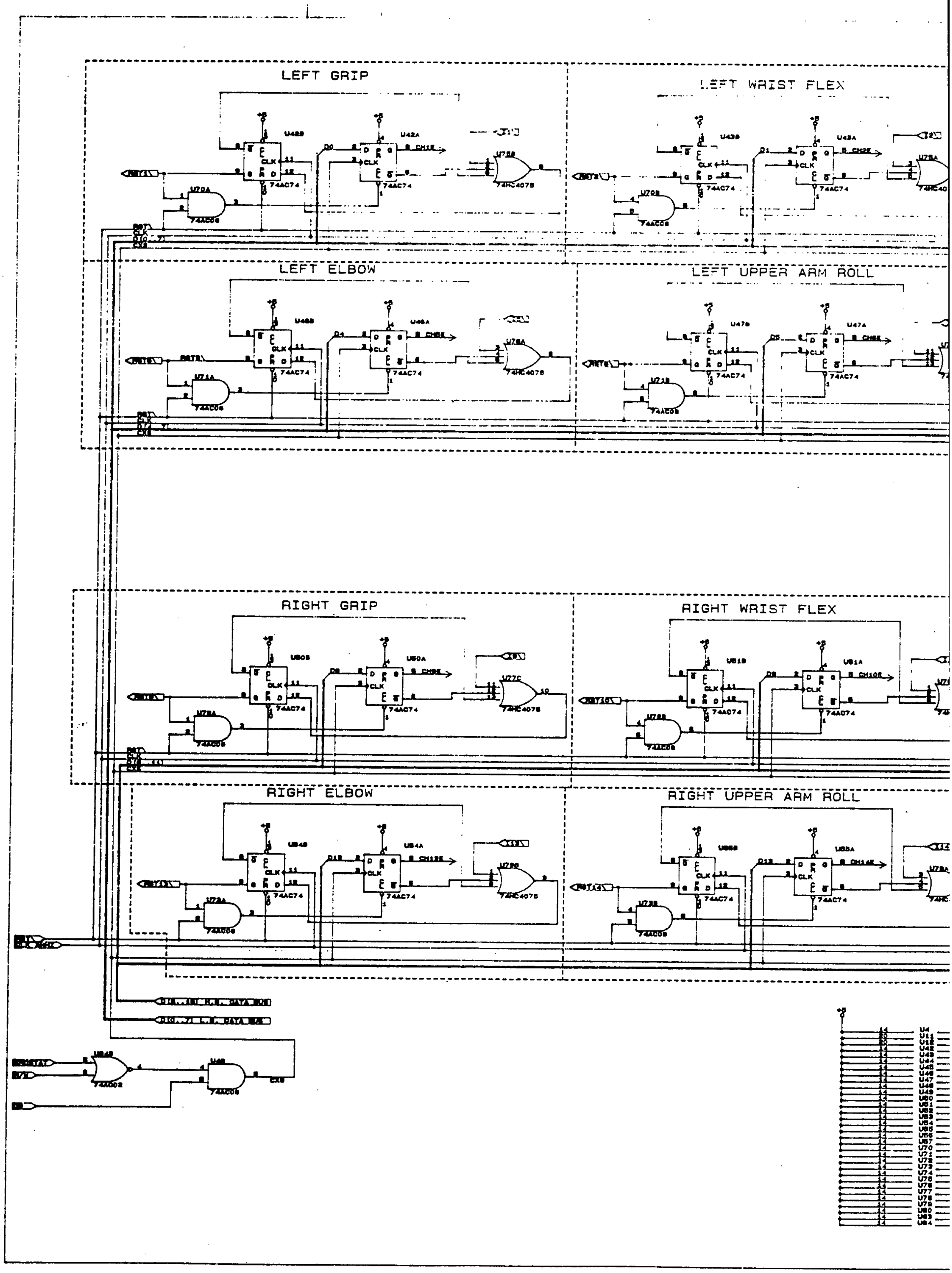

LEFT/RIGHT ENCODER STATUS REGISTER
LEFT/RIGHT ENCODER RECEIVER
128K SRAM AND 16K EPROM
GRIPPER SWITCHES
COUNTER BOARD LAYOUT
MBA PARTS LIST

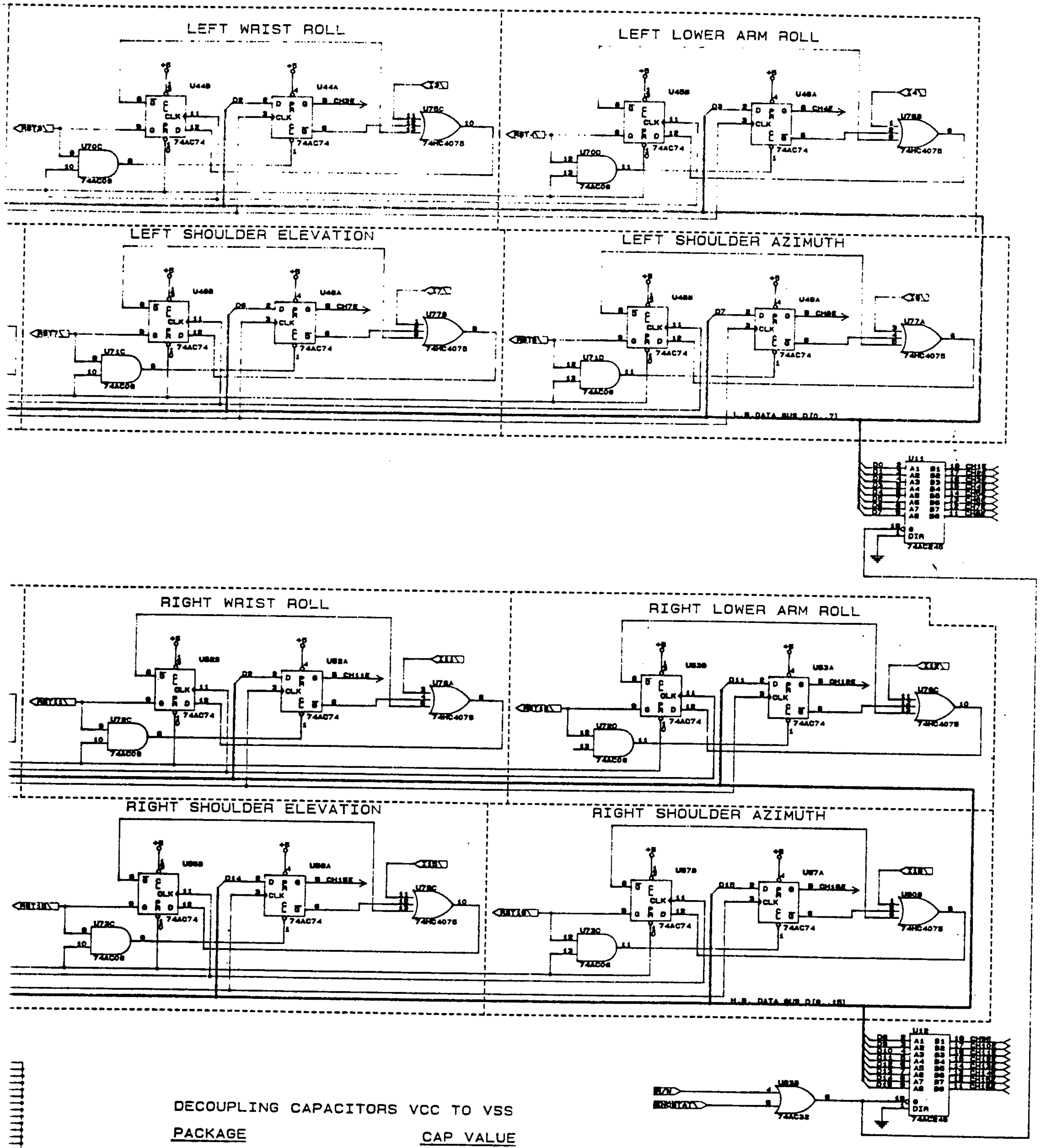

RECEIVER
COUNTER, GRIPPER LED, FOUR PHASE CLK
PROC., SERIAL I/O, SERB BATT., HALT/RESET
PROCESSOR BOARD LAYOUT
ENCODER WIRING/MEMORY MAP

FILE "MBA.@04"

SYSTEMS RESEARCH LABS.		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: DEC. 8, 1989	SHEET 1 OF 11	

2


LEFT GRIP
LEFT WRIST FLEX
LEFT ELBOW
LEFT UPPER ARM ROLL
RIGHT GRIP
RIGHT WRIST FLEX
RIGHT ELBOW
RIGHT UPPER ARM ROLL
U42B
U42A
74AC74
U75B
74HC4075
U70A
74AC08
CH1E
D0
U43B
U43A
CH2E
D1
U75A
U70B
U46B
U46A
CH5E
D4
U76A
U71A
U47B
U47A
CH6E
D5
U71B
U50B
U50A
CH9E
D8
U77C
U72A
U51B
U51A
CH10E
D9
U72B
U54B
U54A
CH13E
D12
U79B
U73A
U55B
U55A
CH14E
D13
U79A
U73B
RST\
CLK
CX8
D[0..15] H.B. DATA BUS
D[0..7] L.B. DATA BUS
U84B
74AC02
U48
74AC08
CX8
+5
U4
U11
U12
U42
U43
U44
U45
U46
U47
U48
U49
U50
U51
U52
U53
U54
U55
U56
U57
U70
U71
U72
U73
U74
U75
U76
U77
U78
U79
U80
U83
U84



DECOUPLING CAPACITORS VCC TO VSS

PACKAGE	CAP VALUE
14	.02uF
20	.03uF

FILE "MBA1.04"

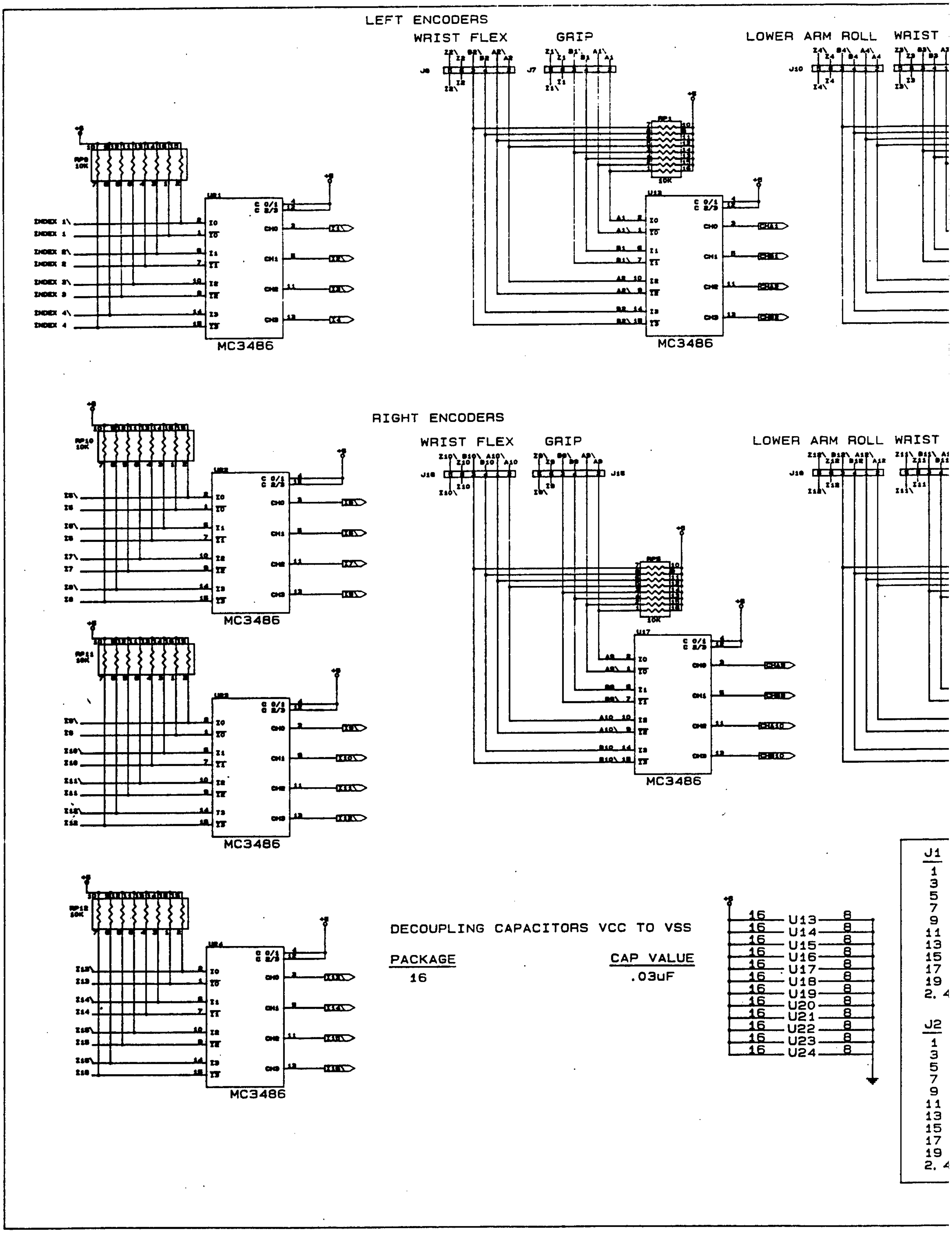
SYSTEMS RESEARCH LABS

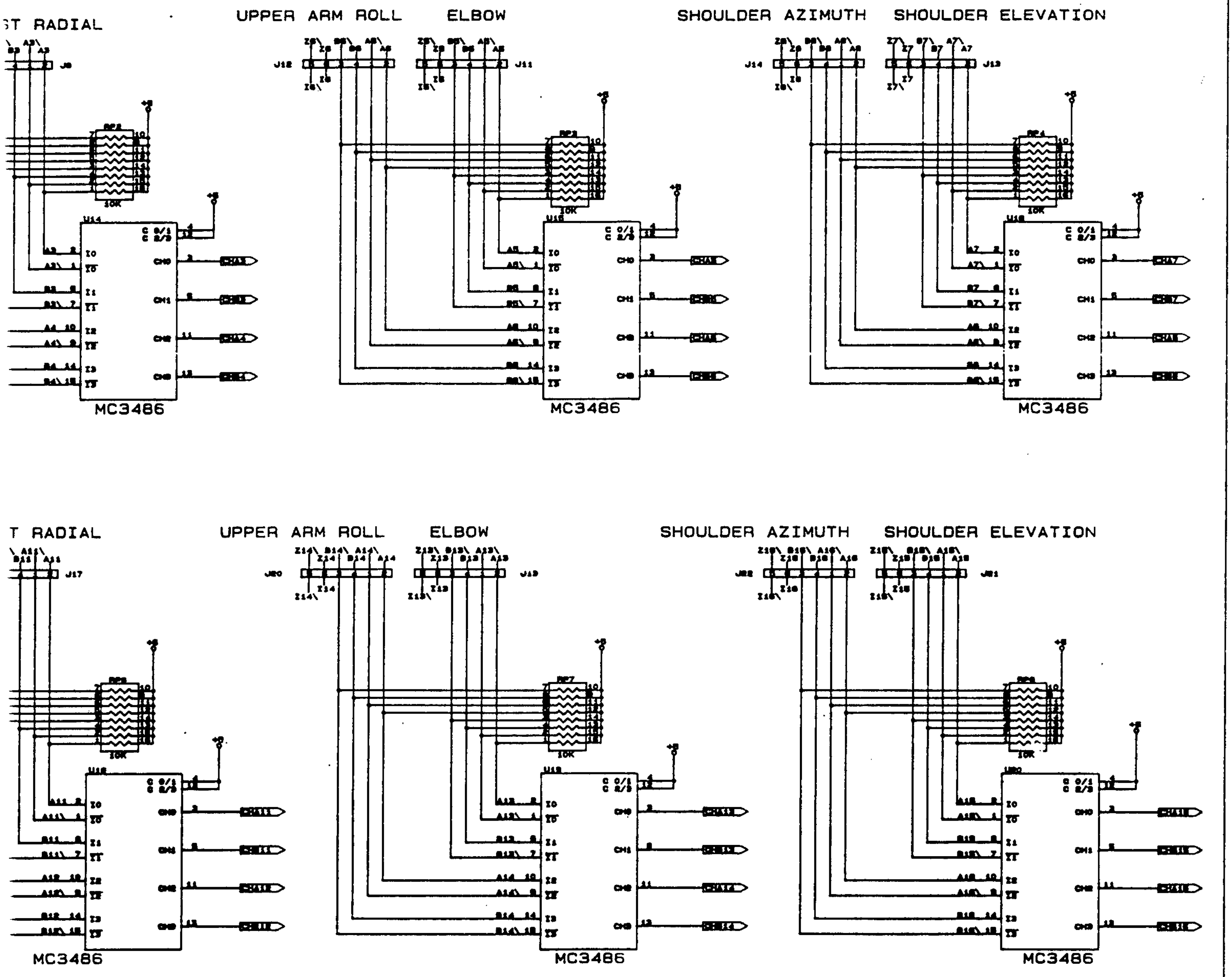
TITLE
MBA EXOSKELETON BACKPACK ELECTRONICS

SIZE D
DOCUMENT NUMBER
ROBTICS TELEPRESENCE

DATE: MARCH 17, 1989 SHEET 2 OF 11

2


LEFT ENCODERS
WRIST FLEX
GRIP
LOWER ARM ROLL
WRIST
RP9
10K
INDEX 1\
INDEX 1
INDEX 2\
INDEX 2
INDEX 3\
INDEX 3
INDEX 4\
INDEX 4
MC3486
RP1
10K
MC3486
RIGHT ENCODERS
WRIST FLEX
GRIP
LOWER ARM ROLL
WRIST
RP10
10K
MC3486
RP11
10K
MC3486
RP12
10K
MC3486
MC3486
DECOUPLING CAPACITORS VCC TO VSS
PACKAGE
16
CAP VALUE
.03uF
16 U13 8
16 U14 8
16 U15 8
16 U16 8
16 U17 8
16 U18 8
16 U19 8
16 U20 8
16 U21 8
16 U22 8
16 U23 8
16 U24 8
J1
1
3
5
7
9
11
13
15
17
19
2, 4
J2
1
3
5
7
9
11
13
15
17
19
2, 4



BOARD TO BOARD CONNECTIONS

1	SIGNAL NAME
	D0
	D1
	D2
	D3
	D4
1	D5
3	D6
5	D7
7	N/C
9	N/C
4, 6...20 - SIGNAL GND	

J3	SIGNAL NAME
1	A1
3	A2
5	A3
7	A4
9	N.C.
11	N.C.
13	N.C.
15	N.C.
17	N.C.
19	N.C.
2, 4, 6...20 - SIGNAL GND	

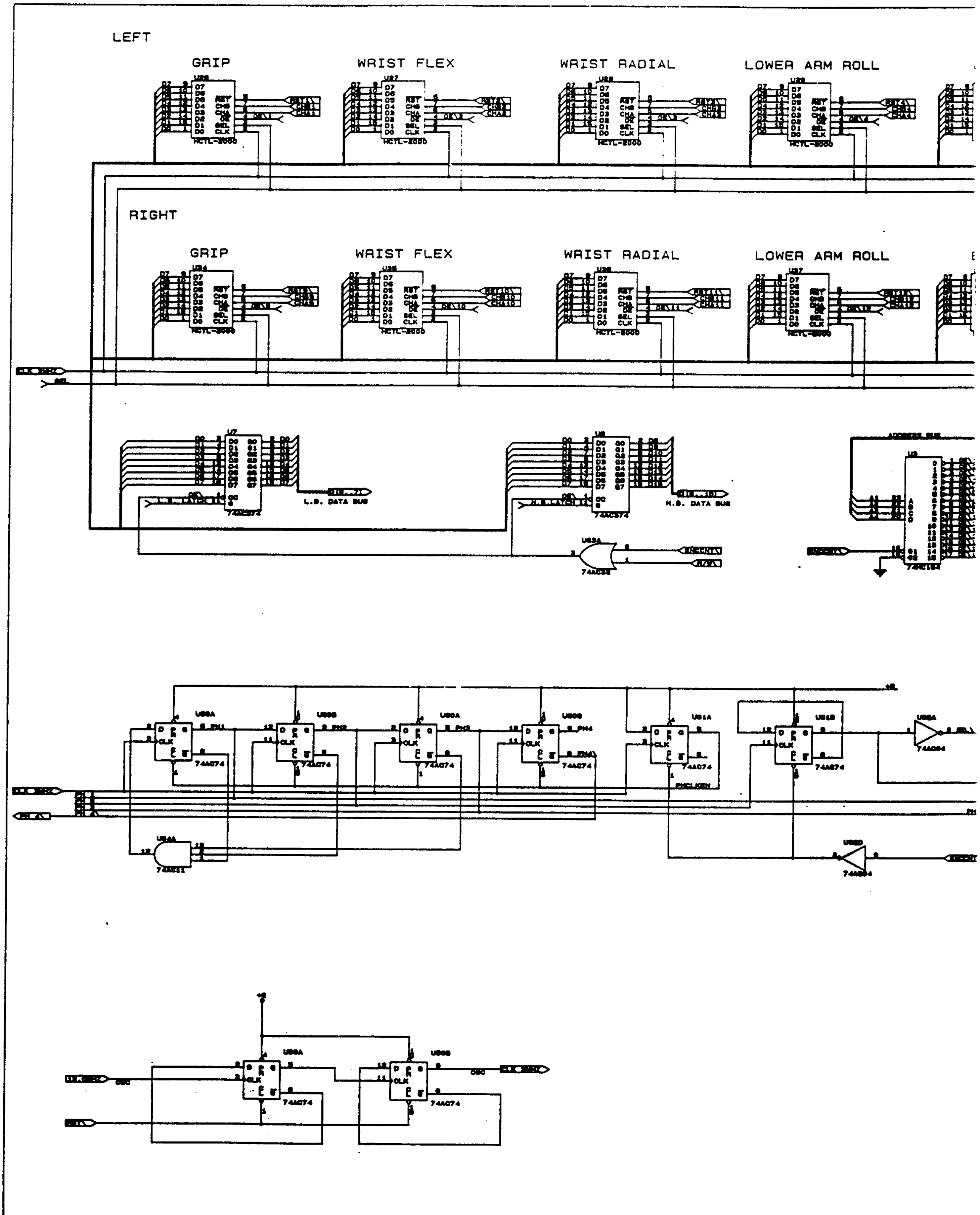
2	SIGNAL NAME
	12MHZ
	R/W\
	D$
	N.C.
	R/W
1	PH4\
3	RST\
5	ENCCNT\
7	ENCSTAT\
9	GRIPLED\
4, 6...20 - SIGNAL GND	

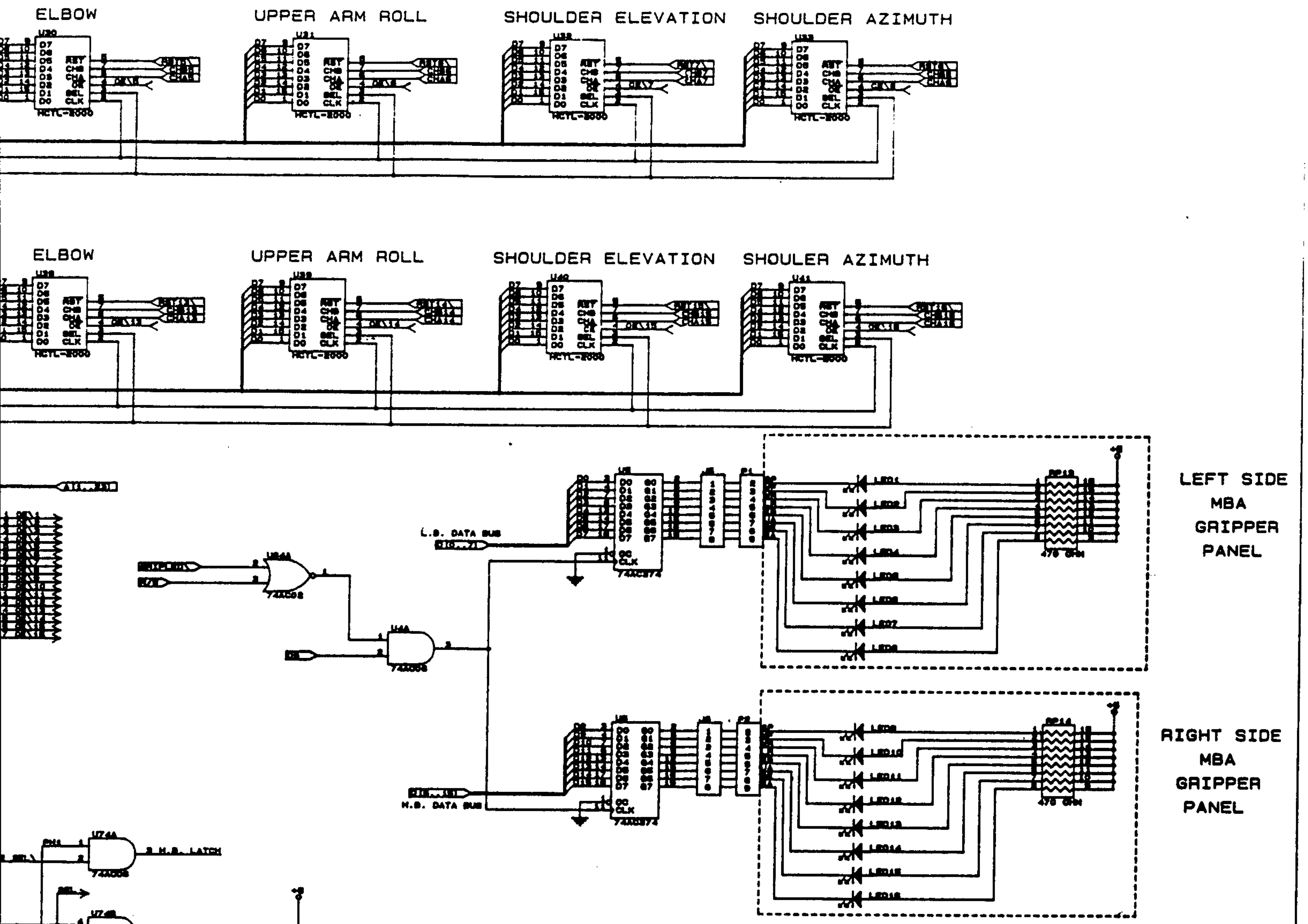
J4	SIGNAL NAME
1	D8
3	D9
5	D10
7	D11
9	D12
11	D13
13	D14
15	D15
17	N.C.
19	N.C.
2, 4, 6...20 - SIGNAL GND	

SYSTEMS RESEARCH LABS		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS		
SIZE D	DOCUMENT NUMBER ROBTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 3 OF 11	

FILE "MBA2.04"

2


LEFT
GRIP
WRIST FLEX
WRIST RADIAL
LOWER ARM ROLL
HCTL-2000
RIGHT
GRIP
WRIST FLEX
WRIST RADIAL
LOWER ARM ROLL
HCTL-2000
CLK 2MHZ
U7
74AC574
L.S. LATCH
L.S. DATA BUS
U8
74AC574
H.S. LATCH
H.S. DATA BUS
74AC32
ADDRESS BUS
74AC154
+5
74AC74
PH1
PH2
PH3
PH4
PHCLKEN
74AC04
SEL\
74AC21
OSC
CLK 2MHZ



+5

24	U3	12
20	U5	10
20	U6	10
20	U7	10
20	U8	10
16	U26	8
16	U27	8
16	U28	8
16	U29	8
16	U30	8
16	U31	8
16	U32	8
16	U33	8
16	U34	8
16	U35	8
16	U36	8
16	U37	8
16	U38	8
16	U39	8
16	U40	8
16	U41	8
14	U58	7
14	U59	7
14	U60	7
14	U61	7
14	U62	7
14	U64	7
14	U74	7

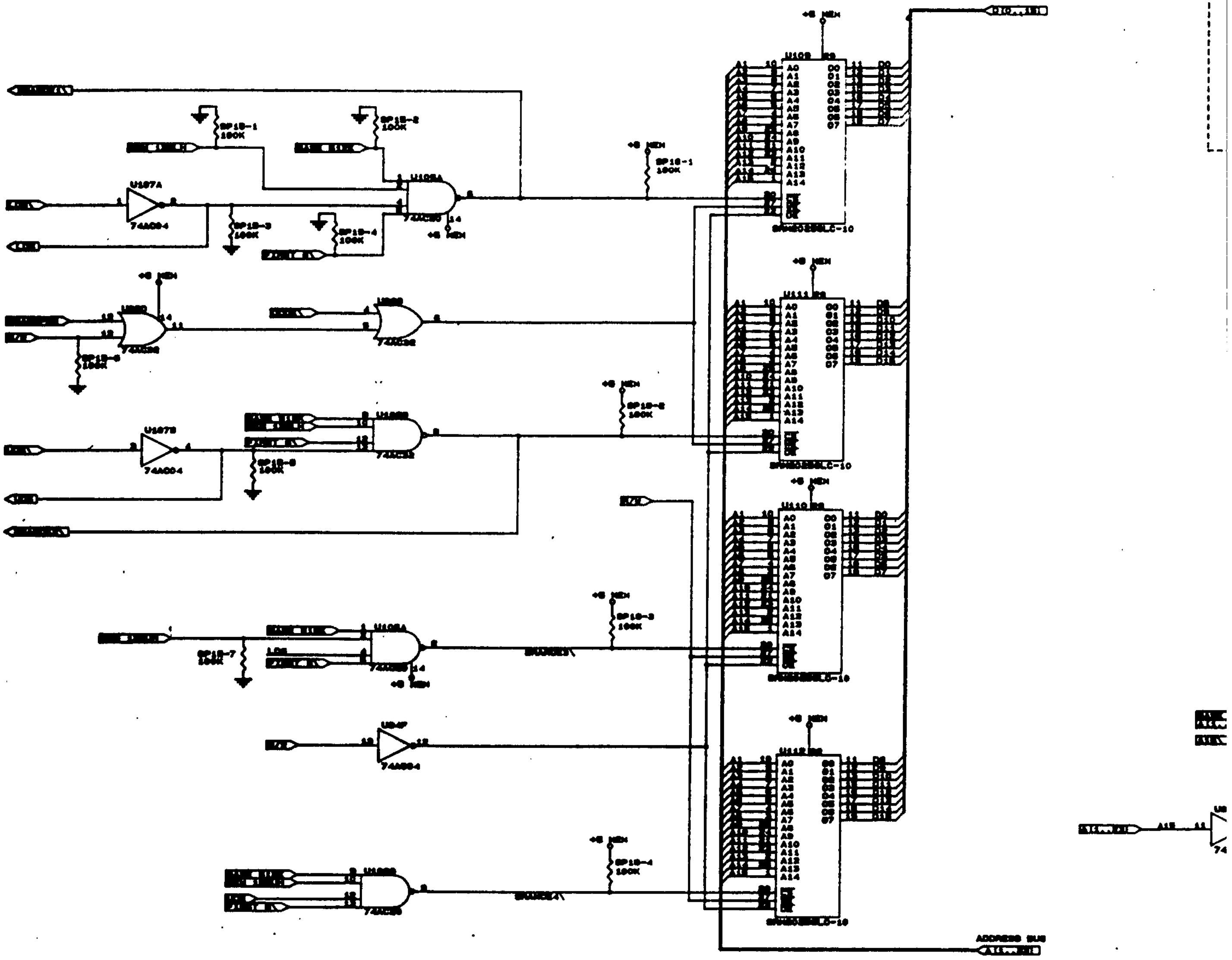
DECOUPLING CAPACITORS VCC TO VSS

PACKAGE	CAP VALUE
24	.03
20	.03
16	.03
14	.02

FILE "MBA3.04"

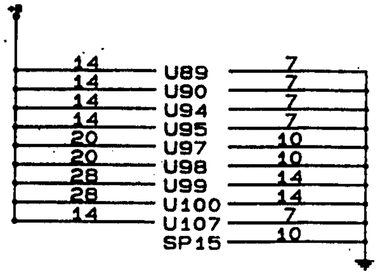
SYSTEMS RESEARCH LABS		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS AND GRIPPER LEDS		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 4 OF 11	

2



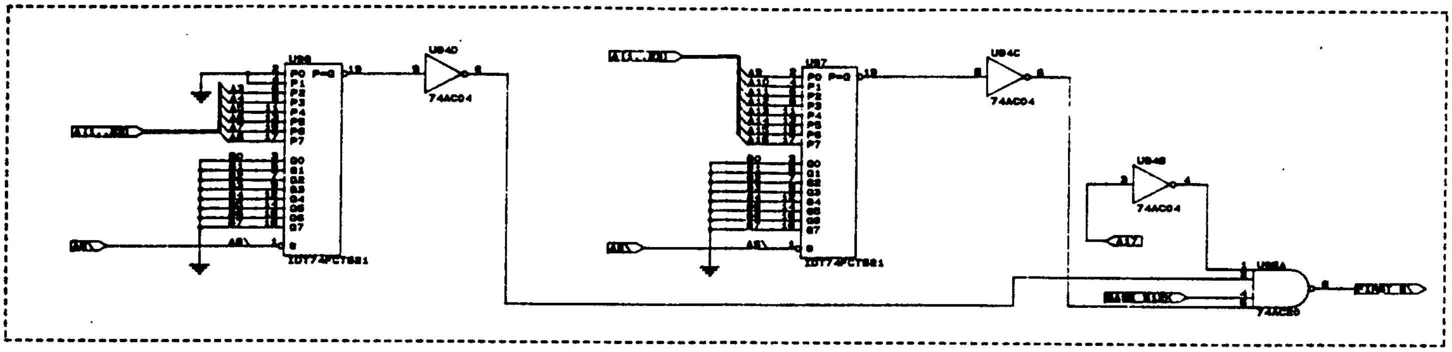
DECOUPLING CAPACITORS VCC TO VSS

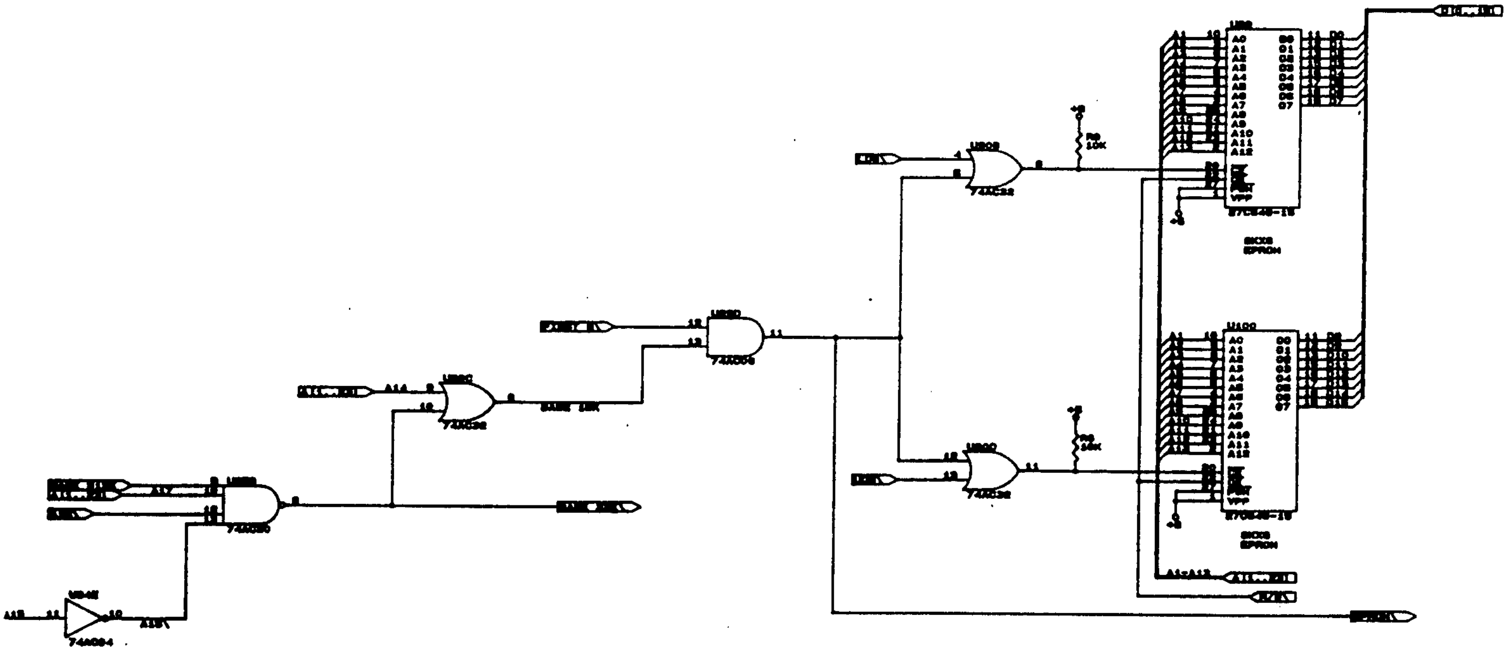
PACKAGE	CAP VALUE
28	.03
20	.03
14	.02

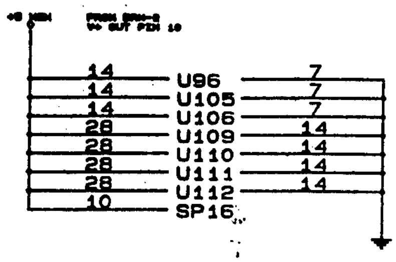


14	U89	7
14	U90	7
14	U94	7
14	U95	7
20	U97	10
20	U98	10
28	U99	14
28	U100	14
14	U107	7
	SP15	10

FIRST EIGHT ADDRESS LOCATIONS DECODE





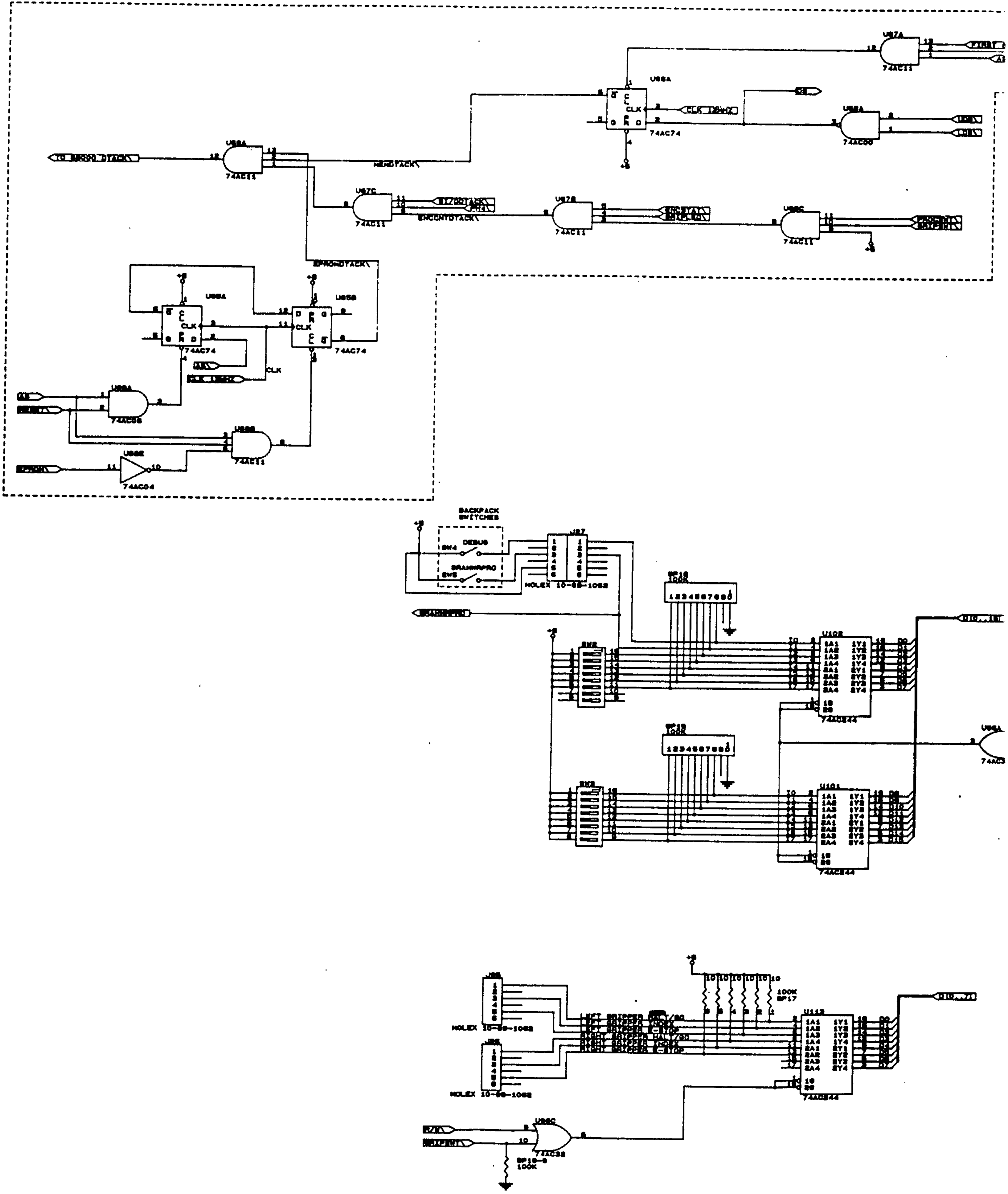


FILE "MBA4.04"

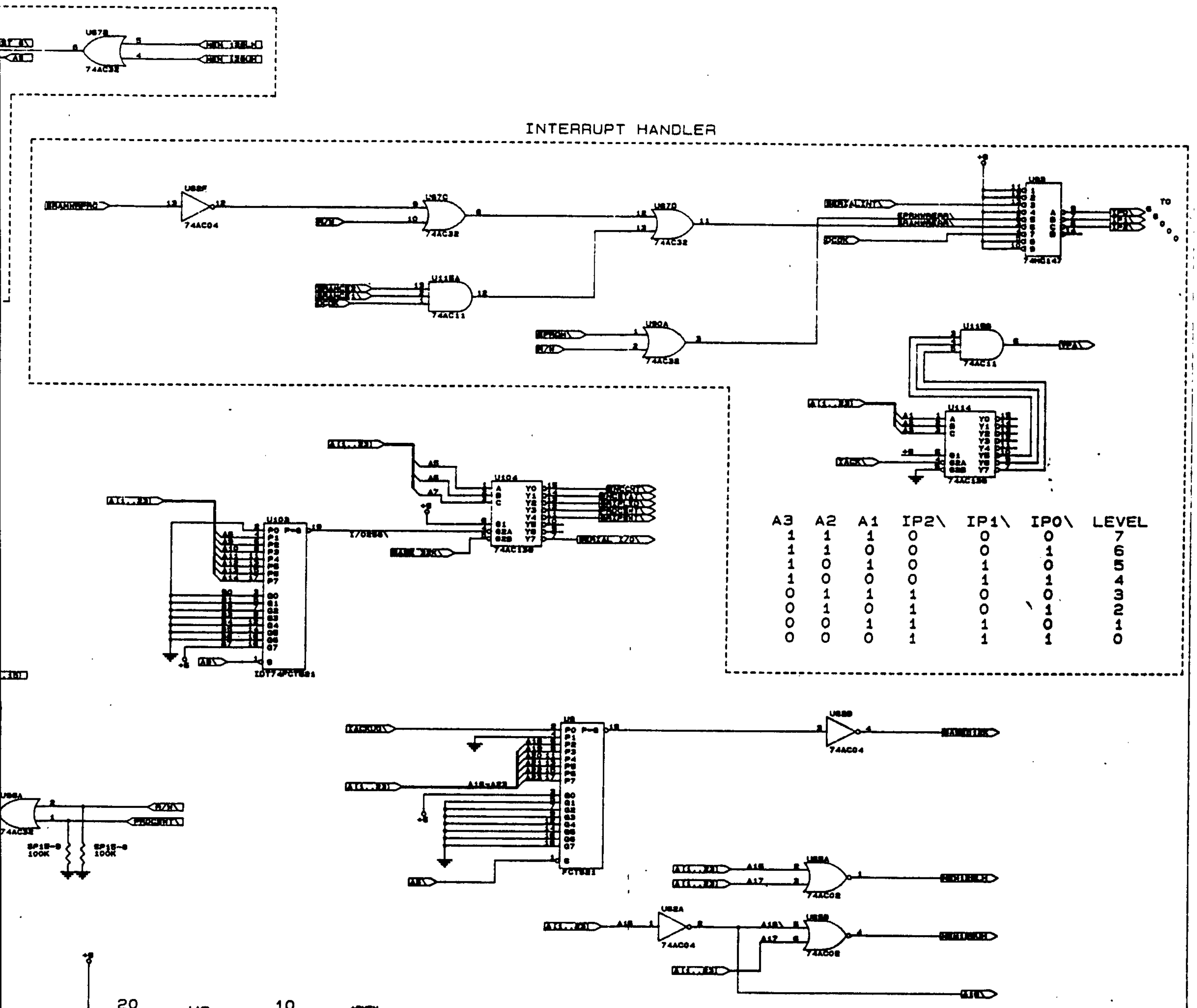
SYSTEMS RESEARCH LABS		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS 128K SRAM AND 16K EPROM		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV A
DATE: MARCH 17, 1989	SHEET 5 OF 11	

2

DTACK GENERATOR



INTERRUPT HANDLER



A3	A2	A1	IP2\	IP1\	IP0\	LEVEL
1	1	1	0	0	0	7
1	1	0	0	0	1	6
1	0	1	0	1	0	5
1	0	0	0	1	1	4
0	1	1	1	0	0	3
0	1	0	1	0	1	2
0	0	1	1	1	0	1
0	0	0	1	1	1	0

+5

20	U9	10
14	U65	7
14	U66	7
14	U67	7
14	U82	7
14	U85	7
14	U86	7
14	U87	7
14	U88	7
14	U89	7
14	U90	7
16	U93	8
14	U96	7
20	U101	10
20	U102	10
20	U103	10
16	U104	8
20	U113	10
16	U114	8
	SP18	10
	SP19	10

+5MEM

14	U115	7

DECOUPLING CAPACITORS VCC TO VSS

PACKAGE	CAP VALUE
20	.03
16	.03
14	.02

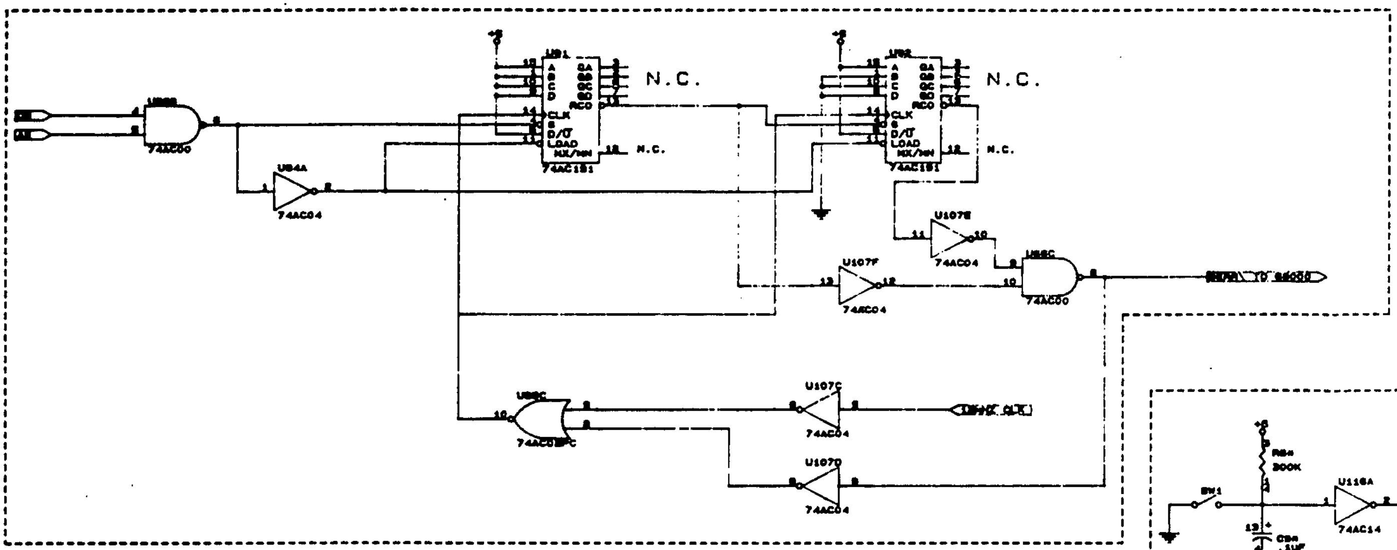
SYSTEMS RESEARCH LABS

TITLE	MBA EXOSKELETON BACKPACK ELECTRONICS (DTACK) (INTERRUPT HANDLER) (ADDRESS DECODE) GRIPPER SWITCHES	
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 6 OF 11	

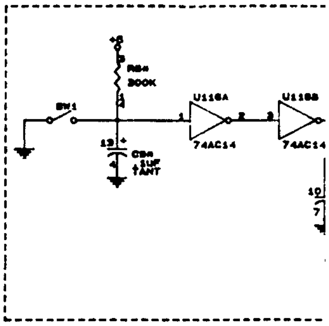
FILENAME: "MBA5.04"

2

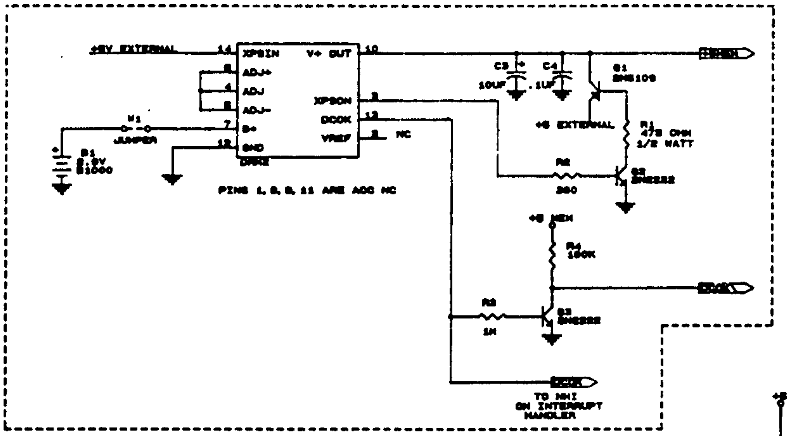
BUS ERROR GENERATOR



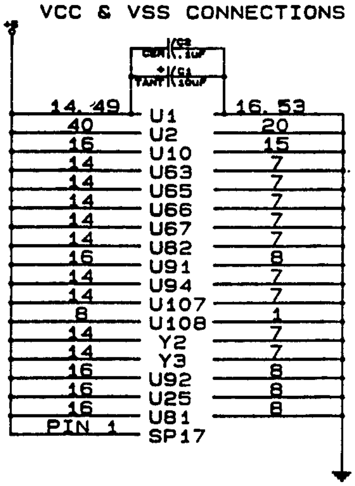
HALT



SRAM BATTERY BACKUP



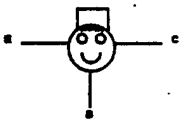
VCC & VSS CONNECTIONS



14, 49	U1	16, 53
40	U2	20
16	U10	15
14	U63	7
14	U65	7
14	U66	7
14	U67	7
14	U82	7
16	U91	8
14	U94	7
14	U107	7
8	U108	1
14	Y2	7
14	Y3	7
16	U92	8
16	U25	8
16	U81	8
PIN 1	SP17	

TOP VIEW

2N2222

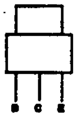


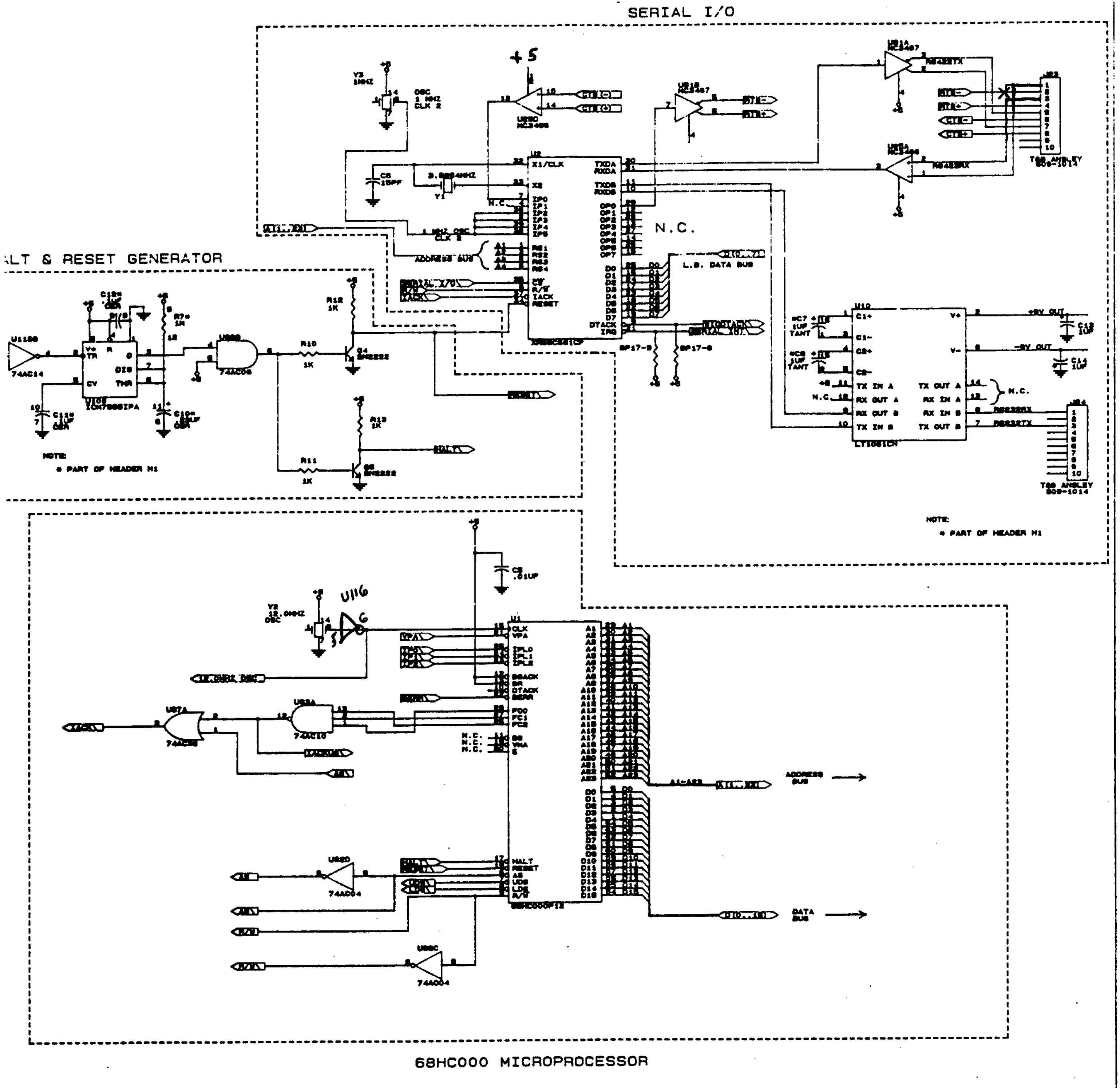
DECOUPLING CAPS VCC TO VSS

PACKAGE	VALUE
16	.03uF
14	.02uF
28	.07uF

FRONT VIEW

2N6109

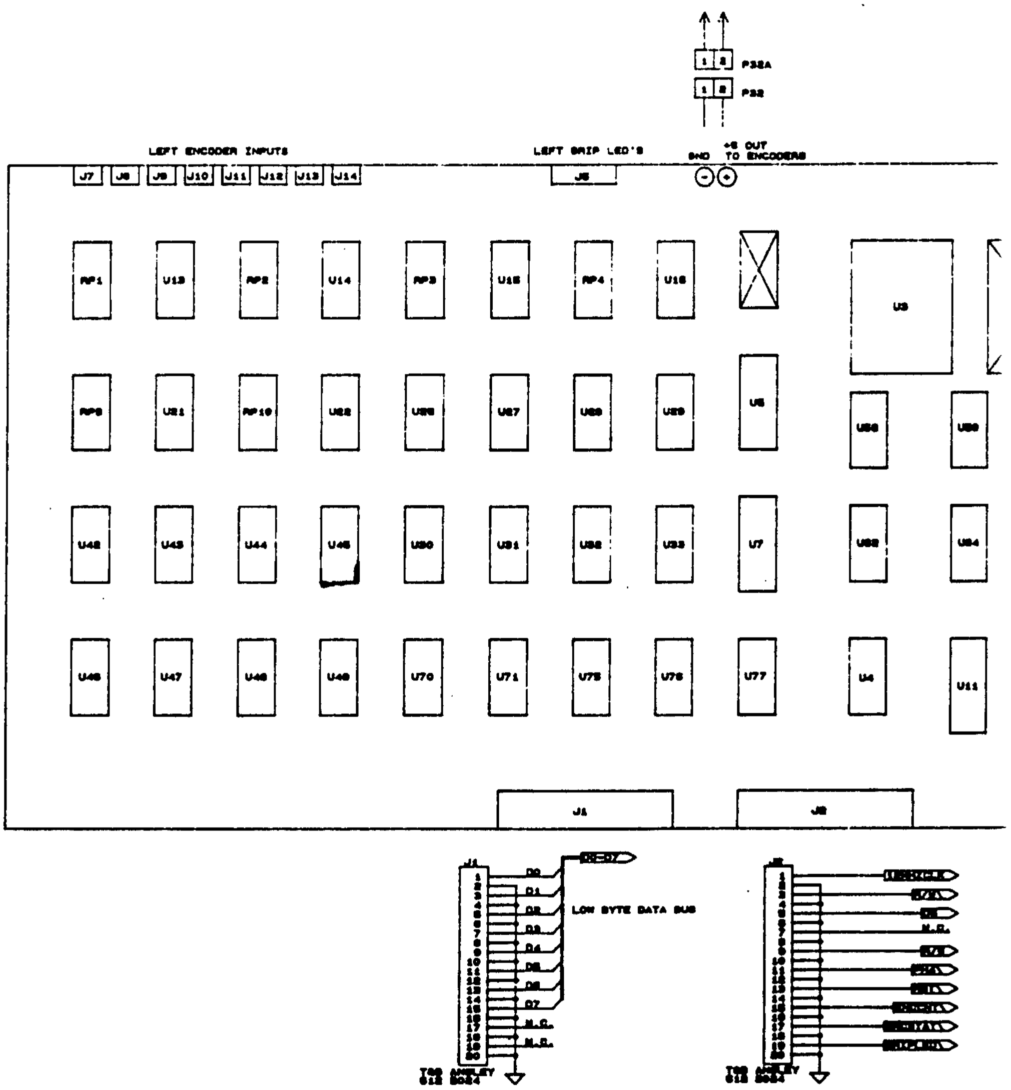


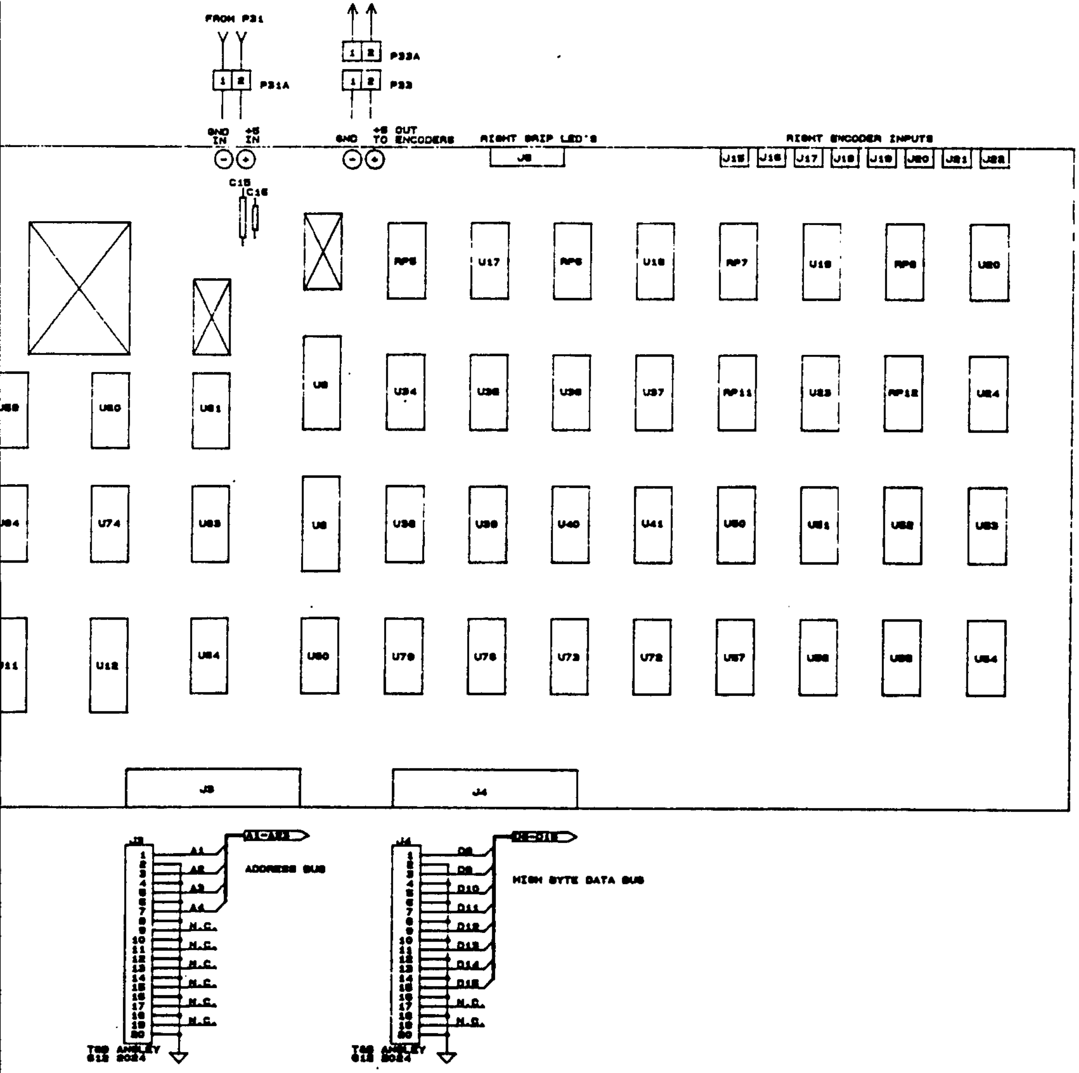


SYSTEMS RESEARCH LABS		
TITLE	MBA EXOSKELETON BACKPACK ELECTRONICS (68HC000) (SERIAL I/O) (BERR) (BATT BACK-UP) (HALT/RESET)	
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 7 OF 11	

FILE "MBA6.04"

2


P32A
P32
LEFT ENCODER INPUTS
LEFT GRIP LED'S
GND
+5 OUT TO ENCODERS
J7
J8
J9
J10
J11
J12
J13
J14
J6
RP1
U13
RP2
U14
RP3
U15
RP4
U16
U3
RP9
U21
RP10
U22
U26
U27
U28
U29
U5
U58
U59
U42
U43
U44
U45
U30
U31
U32
U33
U7
U52
U54
U46
U47
U48
U49
U70
U71
U75
U76
U77
U4
U11
J1
J2
D0-D7
D0
D1
D2
D3
D4
D5
D6
D7
N.C.
N.C.
LOW BYTE DATA BUS
N.C.
T&B ANSLEY
612 2024
T&B ANSLEY
612 2024

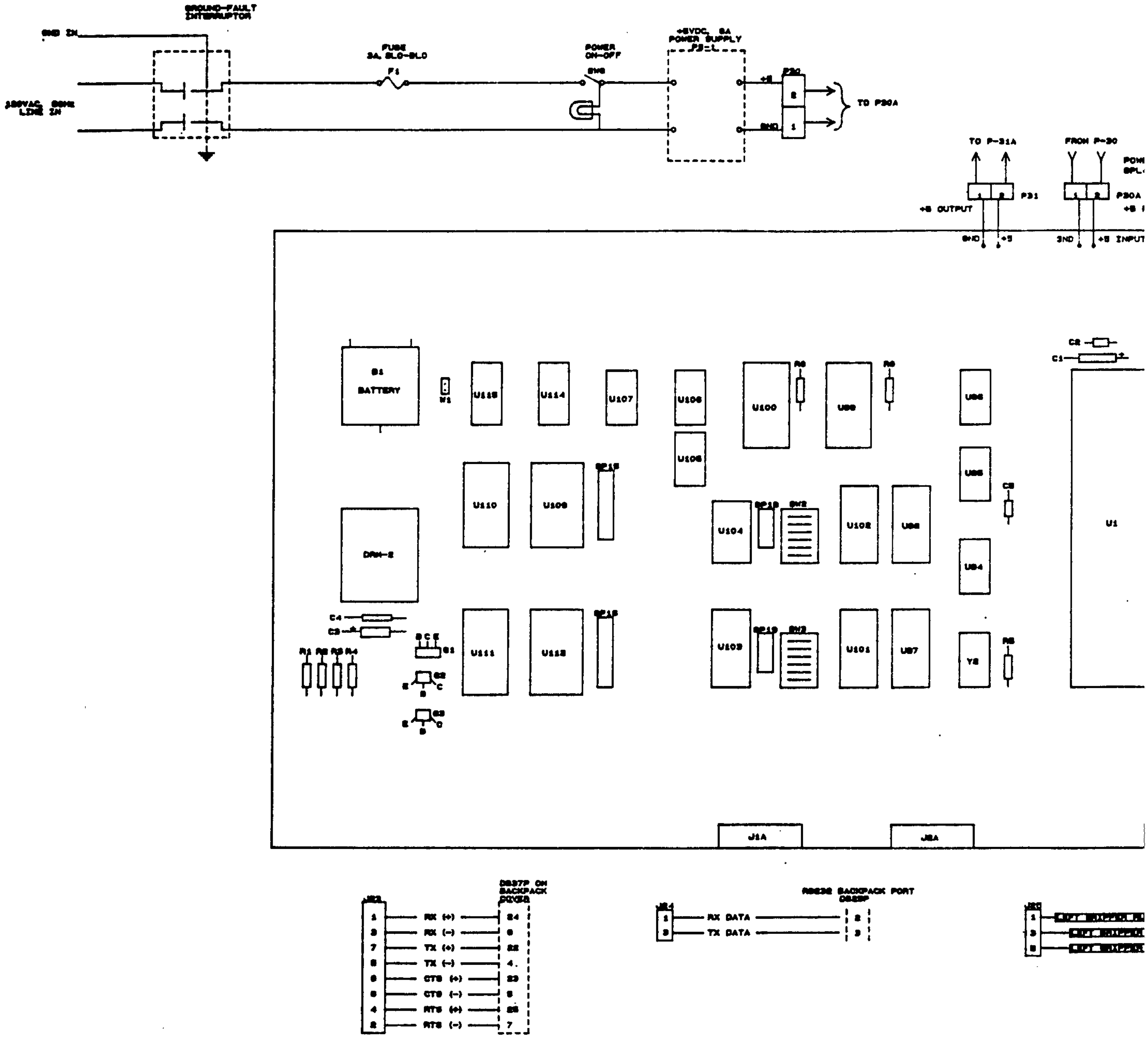


FILE "MBA7.04"

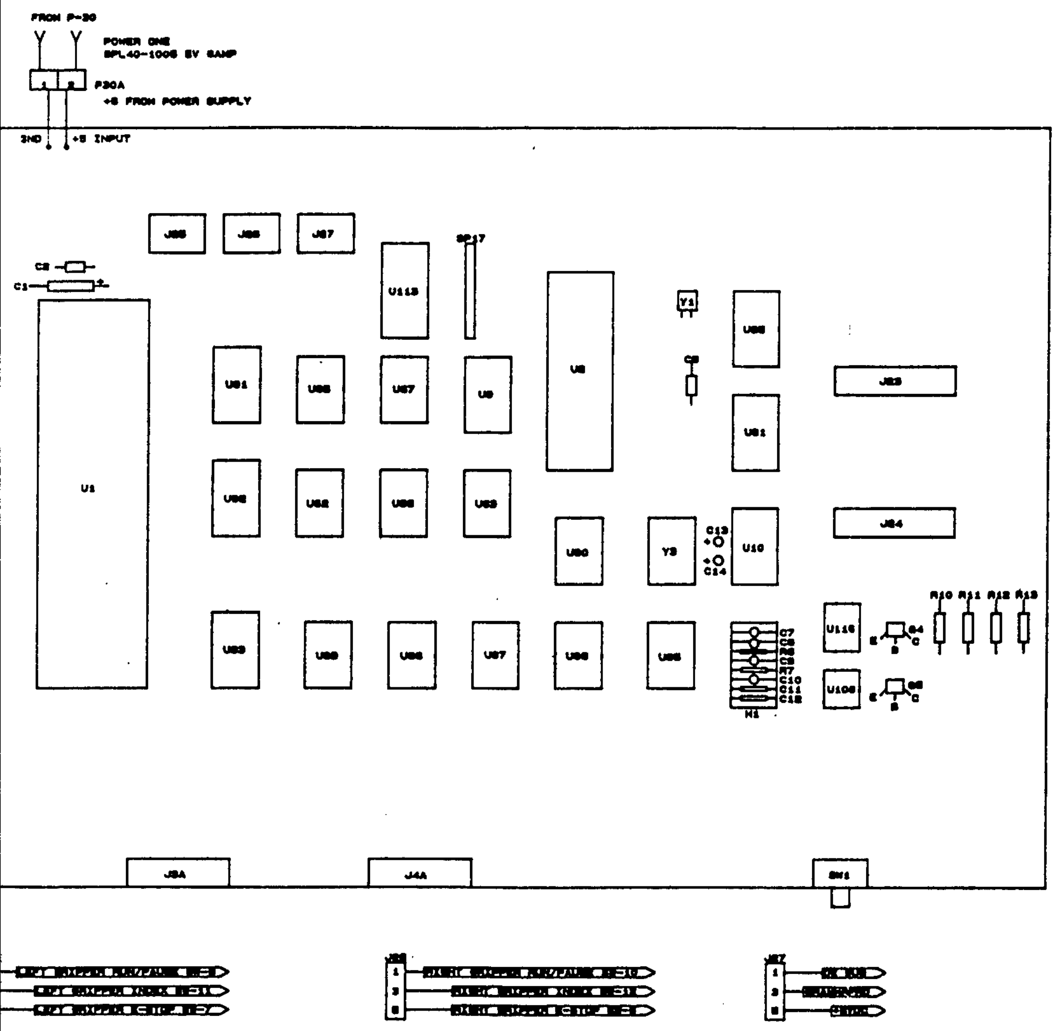
SYSTEMS RESEARCH LABS		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS BOARD LAYOUT		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 8 OF 11	

2

POWER INPUT CIRCUIT



\



FILE "MBA8.04"

SYSTEMS RESEARCH LABS		
TITLE MBA EXOSKELETON BACKPACK ELECTRONICS BOARD LAYOUT		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: MARCH 17, 1989	SHEET 9 OF 11	

2

MANUFACTURER	PART NUMBER	QUANTITY	DESCRIPTION
MOTOROLA	68HC000P12	1	16-/32-BIT MICROPROCESSOR
SIGNETICS	XR68C681CP	1	DUAL ASYNCHRONOUS RECEIVER/TRANSMIT
HEWLETT PACKARD	HCTL-2000	16	QUADRATURE DECODER/COUNTER INTERFAC.
FAIRCHILD	74AC00	1	QUAD 2 INPUT NAND GATE / AVAILABLE
FAIRCHILD	74AC02	2	QUAD 2 INPUT NOR GATE / AVAILABLE G.
FAIRCHILD	74AC04	4	HEX INVERTER / AVAILABLE GATES: U62
FAIRCHILD	74AC08	7	QUAD 2 INPUT AND GATE / AVAILABLE G.
FAIRCHILD	74AC10	1	TRIPLE 3 INPUT NAND GATE / AVAILABL
FAIRCHILD	74AC11	4	TRIPLE 3 INPUT AND GATE / AVAILABLE
FAIRCHILD	74AC14	1	HEX SCHMIDT TRIGGER
FAIRCHILD	74AC20	3	DUAL 4 INPUT NAND
FAIRCHILD	74AC32	5	QUAD 2 INPUT OR GATE / AVAILABLE GA
FAIRCHILD	74AC74	22	DUAL D FLIP-FLOP / AVAILABLE FLIP-F
FAIRCHILD	74AC138	2	1 OF 8 DECODER/DEMULTIPLEXER
MOTOROLA	74HC147	1	DECIMAL- TO - BCD ENCODER
MOTOROLA	74HC154	1	1 OF 16 DECODER/DEMULTIPLEXER
FAIRCHILD	74AC191	2	UP/DOWN COUNTER
FAIRCHILD	74AC244	3	3-STATE OCTAL BUFFER/LINE DRIVER
FAIRCHILD	74AC245	2	OCTAL BIDIRECTIONAL TRANSCEIVER
FAIRCHILD	74AC374	4	OCTAL D-TYPE FLIP-FLOP
MOTOROLA	74HC4075	6	TRIPLE 3 INPUT OR GATE / AVAILABLE (
MOTOROLA	MC3486	13	QUAD RS422 LINE RECEIVER / AVAILABL
MOTOROLA	MC3487	1	QUAD RS422 DRIVER / AVAILABLE PORTS:
LINEAR TECHNOLOGY	LT1081CN	1	RS232 DRIVER/RECEIVER
INTERSIL	ICM7555IPA	1	TIMER
IDT	IDT74FCT521	4	8-BIT IDENTITY COMPARATOR
SMOS SYSTEMS	SRM20256LC-10	4	256K BIT STATIC RAM
HITACHI	27C64G-15	2	8-BIT EPROM
CATALYST RESEARCH	B1000	1	2.8V LITHIUM IODINE BATTERY
CATALYST RESEARCH	DRM-2	1	BATTERY BACKUP MODULE
MOTOROLA	2N6109	1	PNP POWER TRANSISTOR
MOTOROLA	2N2222	4	NPN TRANSISTORS
POWER ONE	SPL40-1005	1	5 VOLT D.C.. 8 AMP POWER SUPPLY (LO
BOURNS	4116R-001-103	12	10K OHM 1%. RESISTOR PACKS
BOURNS	1-104G	4	100K SIP RESISTOR PACKS
BOURNS	4116R-001-471	2	470 OHM DIP RESISTOR PACKS (LOCATED
ALLEN BRADLEY	RCR07G4750JS	1	475 OHM RESISTOR. 1/4W 1%
ALLEN BRADLEY	RCR07G361JS	1	360 OHM RESISTOR. 1/4W 5%
ALLEN BRADLEY	RCR07G105JS	3	1M OHM RESISTOR 1/4W 1%
ALLEN BRADLEY	RCR07G104JS	3	100K RESISTORS. 1/4W 5%
ALLEN BRADLEY	RCR07G681JS	1	680 OHM RESISTOR. 1/4W 5%
ALLEN BRADLEY	RCR07G103JS	4	10K RESISTORS. 1/4W 5%
KEMET	T350E106K025AS	2	10uF CAPACITORS. 25V TANT.
KEMET	C320C104K5R5CA	5	.1uF CAPACITORS. 25V CERAMIC
KEMET	C330C103K1G5CA	1	.01uF CAPACITOR. 50V CERAMIC
SPRAGUE	30GAQ15	1	15pF CAPACITOR. 1KV CERAMIC
KEMET	T350A105K025AS	4	1uF CAPACITOR. 25V TANT.
SPRAGUE	1990224X9035AA2	2	.22uF CAPACITORS. 25V TANT..
B-D CRYSTAL	BD03686B	1	3.6864 MHZ CLOCK CRYSTAL
SARONIX	NCC060C-12	1	12.0 MHZ OSCILLATOR
M-TRON	MCO-T1-S3-1.0	1	1 MHZ OSCILLATOR
C&K	8125	1	SPST RESET SWITCH
GRAYHILL	8744	2	DIP SWITCHES
C&K	T101J12Q	1	DEBUG SWITCH
C&K	T101J12Q	1	SRAMWRPRO SWITCH
ARROW HART	1600R1E	1	LIGHTED ROCKER SWITCH (LOCATED ON BA
C&K	7101J12Q	2	LEFT GRIPPER. RIGHT GRIPPER E-STOP S
C&K	T101SH2Q	2	LEFT GRIPPER. RIGHT GRIPPER HALT-GO
C&K	T108SH2Q	2	LEFT GRIPPER. RIGHT GRIPPER IDX SWIT
BUSSMAN	HTB24I	1	FUSEHOLDER (LOCATED ON BACKPACK COVE
DIALIGHT	521-9501-002	16	T-1 3/4 HIGH EFFICIENCY RED LED'S (L
SAMTEC	CA-02-SJC-B	1	BATTERY JUMPER. 2 PIN
T&B ANSLEY	609-2004	8	RIGHT ANGLE PCB MALE HEADER (20 PIN)
MOLEX	22-04-1081	2	8 PIN HEADER
MOLEX	10-89-1062	16	DUAL ROW HEADER
T&B ANSLEY	609-2041CE	8	FEMALE SOCKET TRANSITION CONNECTOR (
MOLEX	10-89-1068	3	RIGHT ANGLE HEADERS
MOLEX	03-06-1023	4	2 PIN HOUSING
MOLEX	03-06-2023	4	2 PIN PLUG
BEI MOTION SYSTEMS	E113-1024-20	12	1024PPR OPTICAL ENCODER
BEI MOTION SYSTEMS	E113-120-20	4	120PPR OPTICAL ENCODER
AMLAN	CDS25L	1	25 PIN MALE D-SUBMINIATURE CONNECTOR
AMLAN	CDS37L	1	37 PIN MALE D-SUBMINIATURE CONNECTOR
MOLEX	03-06-2092	16	9 PIN PLUG
MOLEX	03-06-1092	16	9 PIN RECEPTACLE
MOLEX	02-06-2132	200	MALE CRIMP PINS
MOLEX	02-06-1132	200	FEMALE CRIMP PINS
MOLEX	16-02-0097	100	CRIMP TERMINALS
T&B ANSLEY	609-1014	2	RIGHT ANGLE PCB MALE HEADER (10 PIN)
T&B ANSLEY	609-1041CE	2	FEMALE SOCKET TRANSITION CONNECTOR (

	REFERENCE
	U1
/TRANSMITTER	U2
INTERFACE IC	U26 THRU U41
VAILABLE GATES: U66-D	U66
AILABLE GATES: U65-D, U84-C&D	U65, 84
ATES: U62-B, C, E, F	U62, 82, 94, 107
AILABLE GATES: U80-A&C	U4, 70, 71, 72, 73, 74, 89
AVAILABLE GATES: U63-B&C	U63
AVAILABLE GATES: U64-B&C, U115 - C	U64, 87, 88, 115
	U116
	U95, 105, 106
ILABLE GATES: U83-C&D	U67, 83, 90, 96, 105
E FLIP-FLOP: U86-B	U42 THRU U61, U85, U86
ER	U104, 114
	U93
KER	U3
	U91, 92
DRIVER	U101, 102, 113
EIVER	U11, 12
	U5, 6, 7, 8
AILABLE GATES: U80-A&C	U75, 76, 77, 78, 79, 80
AVAILABLE PORTS: U25 - C&D	U13 THRU U25
LE PORTS: U81 - C&D	U81
	U10
	U108
	U9, 97, 98, 103
	U109, 110, 111, 112
	U99, 100
	B-1
	DRM-2
	Q1
	Q2, 3, 4, 5
PPLY (LOCATED ON BACKPACK COVER)	PS-1
	RP1 THRU RP12
	SP15, SP16, SP17, SP18
(LOCATED IN GRIPPER PANELS)	RP13, RP14
	R1
	R2
	R3, R6*, R7*
	R4, R10, R11
	R5
	R8, R9, R12, R13
	C1, C3, C15
C	C2, C4, C11*, C12*, C16
C	C5
	C6
	*C7, *C8, C13, C14
	C9*, C10*
	Y1
	Y2
	Y3
	SW1
	SW2, SW3
	SW4
	SW5
TED ON BACKPACK COVER)	SW6
E-STOP SWITCH (LOCATED IN GRIPPER PANELS)	SW7, SW8
HALT-GO SWITCH (LOCATED IN GRIPPER PANELS)	SW9, SW10
IDX SWITCH (LOCATED IN GRIPPER PANELS)	SW11, SW12
PACK COVER)	F1
LED'S (LOCATED IN GRIPPER PANELS)	LED 1 THROUGH LED 16
	W1
(20 PIN)	J1, J2, J3, J4, J1A, J2A, J3A, J4A
	J5, J6
	J7 THRU J22
NECTOR (20 PIN)	INTERCONNECT CABLE
	J25, J26, J27
	P30, P31, P32, P33
	P30A, P31A, P32A, P33A
	ENCODERS FOR SA, SE, EB, WF, WR, GP
	ENCODERS FOR UR, LR
ONNECTOR	RS232 PORT
ONNECTOR	RS422 PORT
	ENCODER CONNECTORS
	ENCODER CONNECTORS
	CONNECTOR PINS
	CONNECTOR PINS
	CONNECTOR PINS
(10 PIN)	J23, J24
NECTOR (10 PIN)	INTERCONNECT CABLE

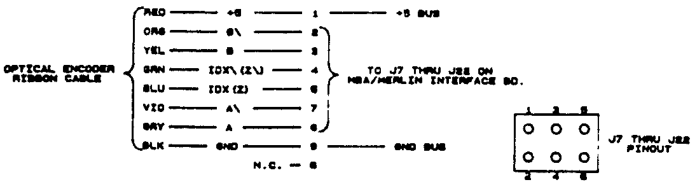
*CAPACITORS C7 THRU C12 LOCATED ON HEADER H1

SYSTEMS RESEARCH LABS		
TITLE MBA PARTS LIST		
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: AUGUST 4, 1989	SHEET 10 OF 11	

FILE "MBA9.04"

2

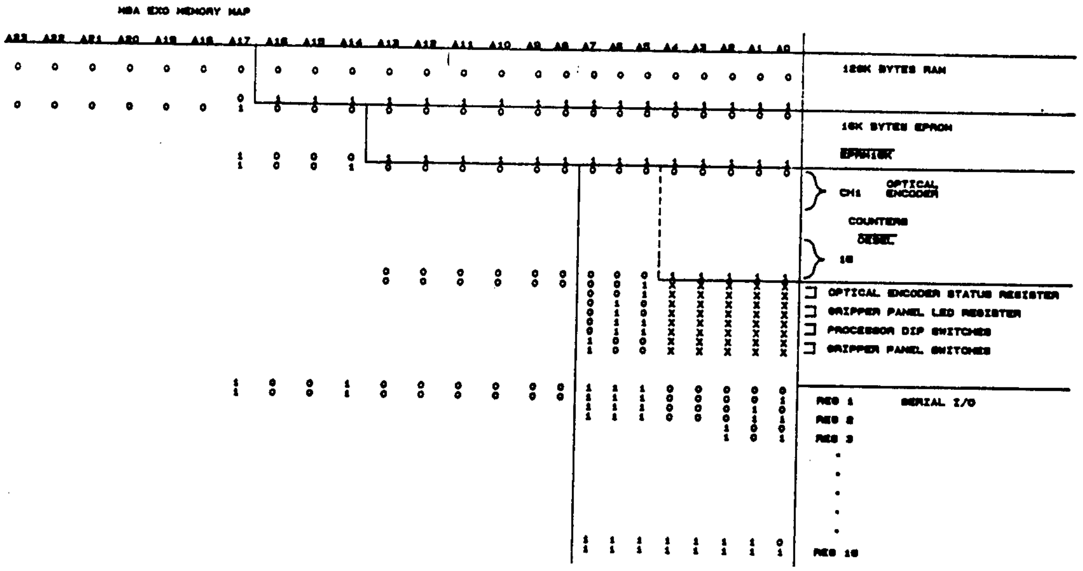
BEI MODEL E113
ENCODER CONNECTIONS



LEFT SIDE ENCODER OUTPUTS

MBA/MERLIN INTERFACE BD.

ENCODER:	DESG:	OUTPUTS:	INPUT CONNECTOR:	TO:
1	LEFT GRIP (GP)	A1\	J7 - 1	U13-1
		A1	2	U13-2
		B1\	3	U13-7
		B1	4	U13-6
		Z1\	5	U21-2
		Z1	6	U21-1
2	LEFT WRIST FLEX (WF)	A2\	J8 - 1	U13-9
		A2	2	U13-10
		B2\	3	U13-15
		B2	4	U13-14
		Z2\	5	U21-6
		Z2	6	U21-7
3	LEFT WRIST RADIAL (WR)	A3\	J9 - 1	U14-1
		A3	2	U14-2
		B3\	3	U14-7
		B3	4	U14-6
		Z3\	5	U21-10
		Z3	6	U21-9
4	LEFT LOWER ARM ROLL (LA)	A4\	J10 - 1	U14-9
		A4	2	U14-10
		B4\	3	U14-15
		B4	4	U14-14
		Z4\	5	U21-14
		Z4	6	U21-15
5	LEFT ELBOW (EB)	A5\	J11 - 1	U15-1
		A5	2	U15-2
		B5\	3	U15-7
		B5	4	U15-6
		Z5\	5	U22-2
		Z5	6	U22-1
6	LEFT UPPER ARM ROLL (UA)	A6\	J12 - 1	U15-9
		A6	2	U15-10
		B6\	3	U15-15
		B6	4	U15-14
		Z6\	5	U22-6
		Z6	6	U22-7
7	LEFT SHOULDER ELEVATION (SE)	A7\	J13 - 1	U16-1
		A7	2	U16-2
		B7\	3	U16-7
		B7	4	U16-6
		Z7\	5	U22-10
		Z7	6	U22-9
8	LEFT SHOULDER AZIMITH (SA)	A8\	J14 - 1	U16-9
		A8	2	U16-10
		B8\	3	U16-15
		B8	4	U16-14
		Z8\	5	U22-14
		Z8	6	U22-15
9	RIGHT GRIP (GP)	A9\	J15 - 1	U17-1
		A9	2	U17-2
		B9\	3	U17-7
		B9	4	U17-6
		Z9\	5	U23-2
		Z9	6	U23-1
10	RIGHT WRIST FLEX (WF)	A10\	J16 - 1	U17-9
		A10	2	U17-10
		B10\	3	U17-15
		B10	4	U17-14
		Z10\	5	U23-6
		Z10	6	U23-7
11	RIGHT WRIST RADIAL (WR)	A11\	J17 - 1	U18-1
		A11	2	U18-2
		B11\	3	U18-7
		B11	4	U18-6
		Z11\	5	U23-10
		Z11	6	U23-9
12	RIGHT LOWER ARM ROLL (LA)	A12\	J18 - 1	U18-9
		A12	2	U18-10
		B12\	3	U18-15
		B12	4	U18-14
		Z12\	5	U23-14
		Z12	6	U23-15
13	RIGHT ELBOW (EB)	A13\	J19 - 1	U19-1
		A13	2	U19-2
		B13\	3	U19-7
		B13	4	U19-6
		Z13\	5	U24-2
		Z13	6	U24-1
14	RIGHT UPPER ARM ROLL (UA)	A14\	J20 - 1	U19-9
		A14	2	U19-10
		B14\	3	U19-15
		B14	4	U19-14
		Z14\	5	U24-6
		Z14	6	U24-7
15	RIGHT SHOULDER ELEVATION (SE)	A15\	J21 - 1	U20-1
		A15	2	U20-2
		B15\	3	U20-7
		B15	4	U20-6
		Z15\	5	U24-10
		Z15	6	U24-9
16	RIGHT SHOULDER AZIMUTH (SA)	A16\	J22 - 1	U20-9
		A16	2	U20-10
		B16\	3	U20-15
		B16	4	U20-14
		Z16\	5	U24-14
		Z16	6	U24-15



SYSTEMS RESEARCH LABS		
TITLE	MBA BACKPACK ELECTRONICS ENCODER WIRING / MEMORY MAP	
SIZE D	DOCUMENT NUMBER ROBOTICS TELEPRESENCE	REV
DATE: AUGUST 4, 1989	SHEET 11 OF 11	

FILE "MBA10.04"

2

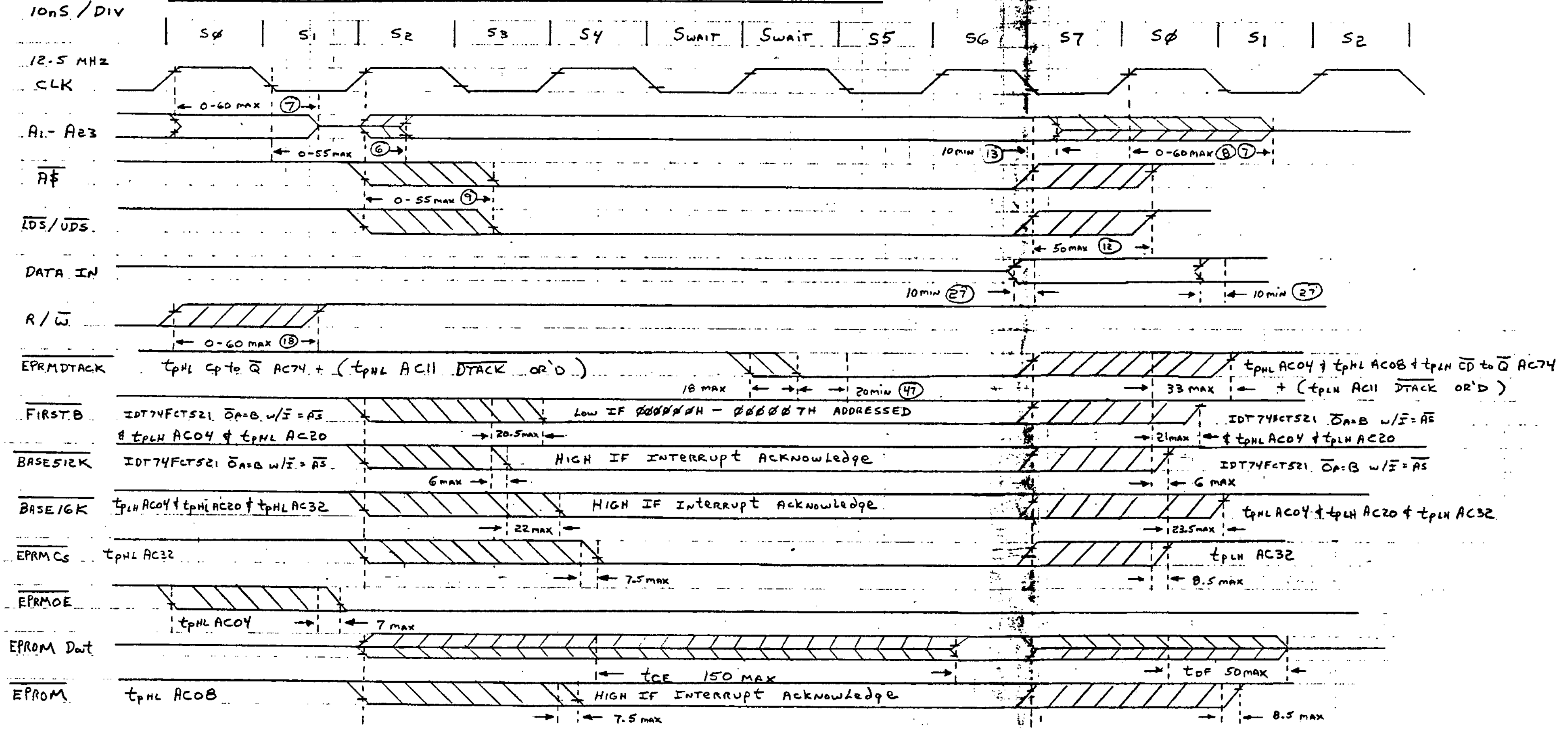
This page intentionally left blank.

9.0 Appendix A: MBAssociates Backpack

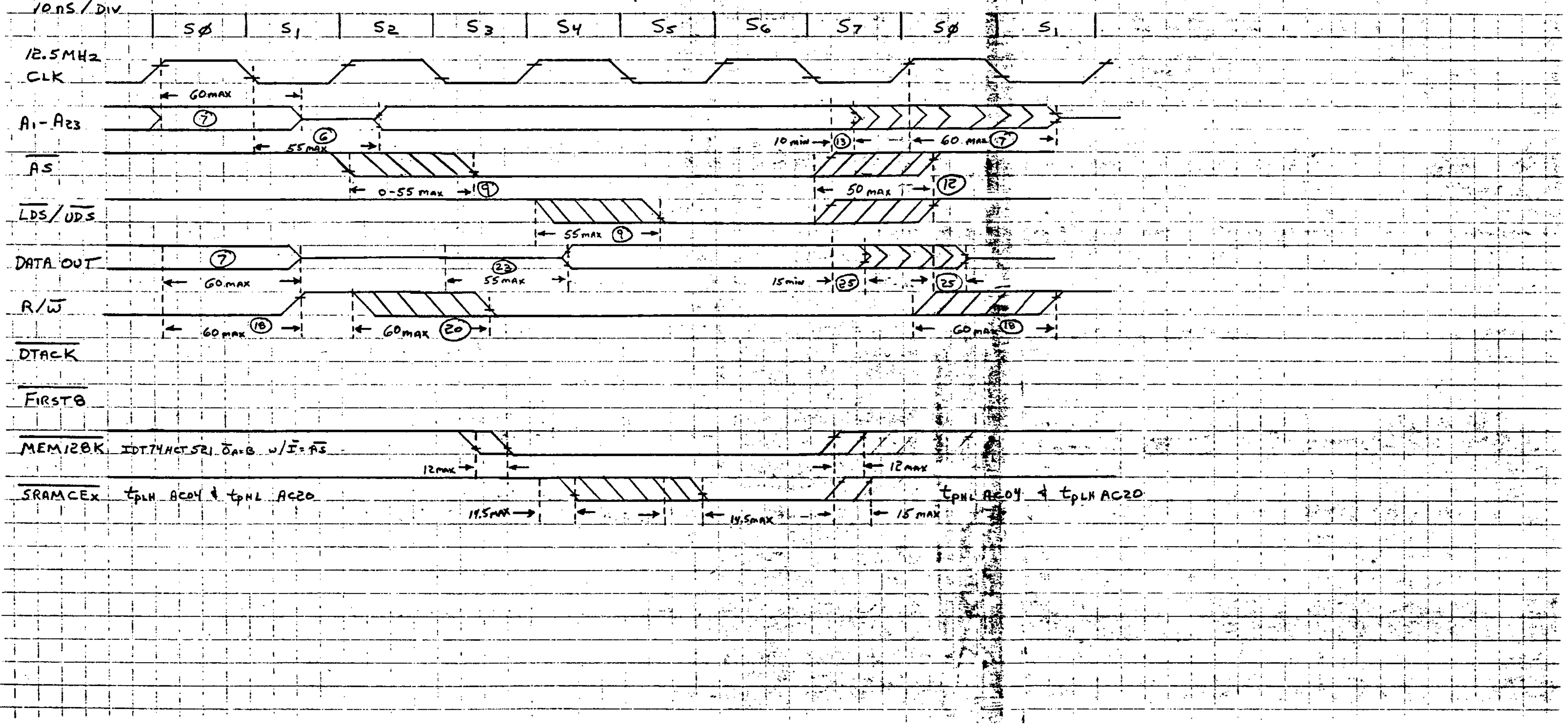
9.2 Timing Diagrams

This page intentionally left blank.

MBA EXO 16K EPROM WAIT State READ CYCLE 9/3/88 Page 1 of 1

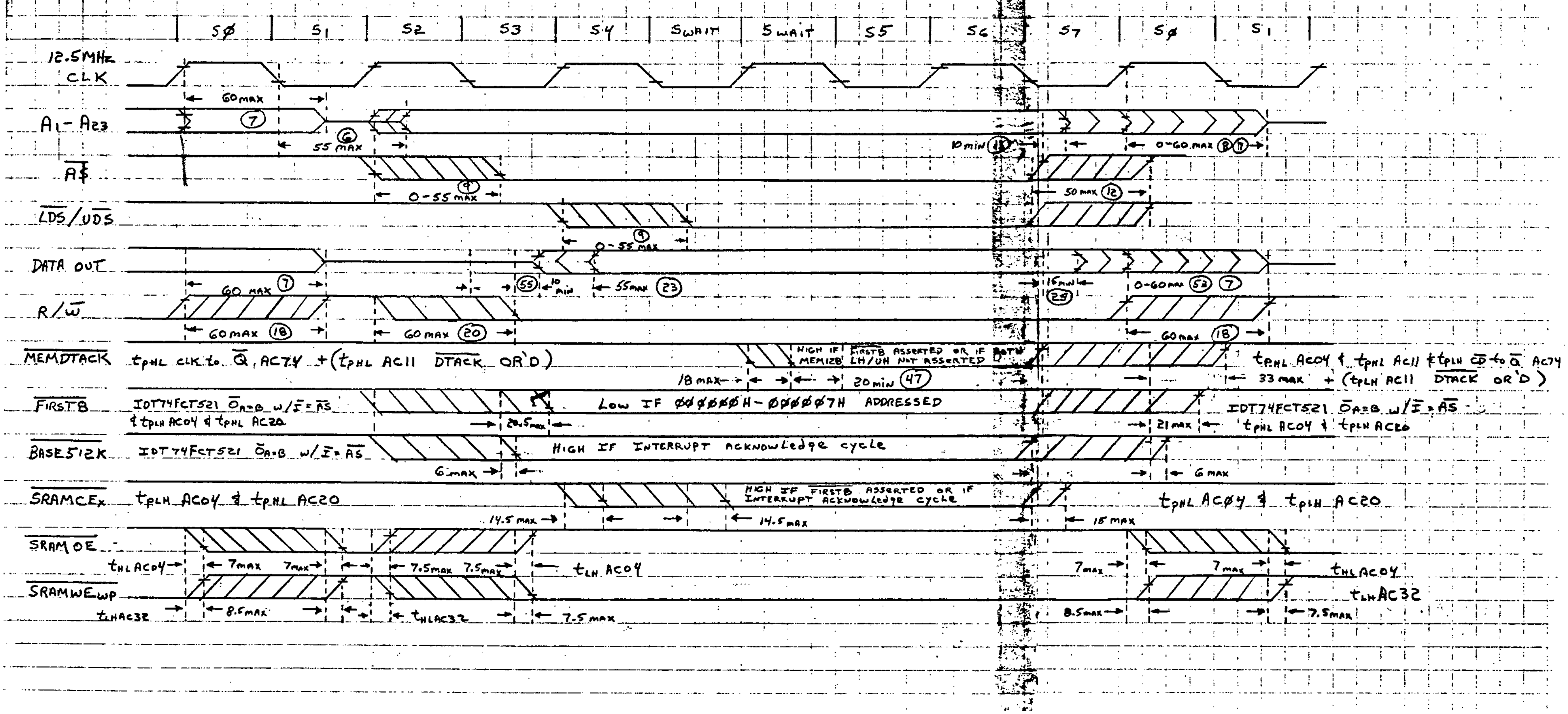

10nS / DIV
Sø | S1 | S2 | S3 | S4 | SWAIT | SWAIT | S5 | S6 | S7 | Sø | S1 | S2
12.5 MHz CLK
A1 - A23
AS
LDS / UDS
DATA IN
R / W
EPRMDTACK
FIRST.B
BASE512K
BASE16K
EPRM CS
EPRMOE
EPROM Dout
EPROM
0-60 max 7
0-55 max 6
0-55 max 9
0-60 max 18
10 min 13
0-60 max 8 7
50 max 12
10 min 27
10 min 27
tPHL CP to Q AC74 + (tPHL AC11 DTACK OR'D)
18 max
20 min 47
tPHL AC04 & tPHL AC08 & tPLH CD to Q AC74
33 max
+ (tPLH AC11 DTACK OR'D)
IDT74FCT521 OA=B w/I = AS
& tPLH AC04 & tPHL AC20
20.5 max
Low IF ØØØØØØH - ØØØØØ7H ADDRESSED
IDT74FCT521 OA=B w/I = AS
21 max
& tPHL AC04 & tPLH AC20
IDT74FCT521 OA=B w/I = AS
6 max
HIGH IF INTERRUPT ACKNOWLEDGE
IDT74FCT521 OA=B w/I = AS
6 max
tPLH AC04 & tPHL AC20 & tPHL AC32
22 max
HIGH IF INTERRUPT ACKNOWLEDGE
tPHL AC04 & tPLH AC20 & tPLH AC32
23.5 max
tPHL AC32
7.5 max
tPLH AC32
8.5 max
tPHL AC04
7 max
tCE 150 MAX
tDF 50 max
tPHL AC08
HIGH IF INTERRUPT ACKNOWLEDGE
7.5 max
8.5 max

MBA EXO 128K SRAM WRITE Cycle 5 August 88 Page 1 of 1



MBA EXO 128K SRAM WAIT STATE WRITE CYCLE

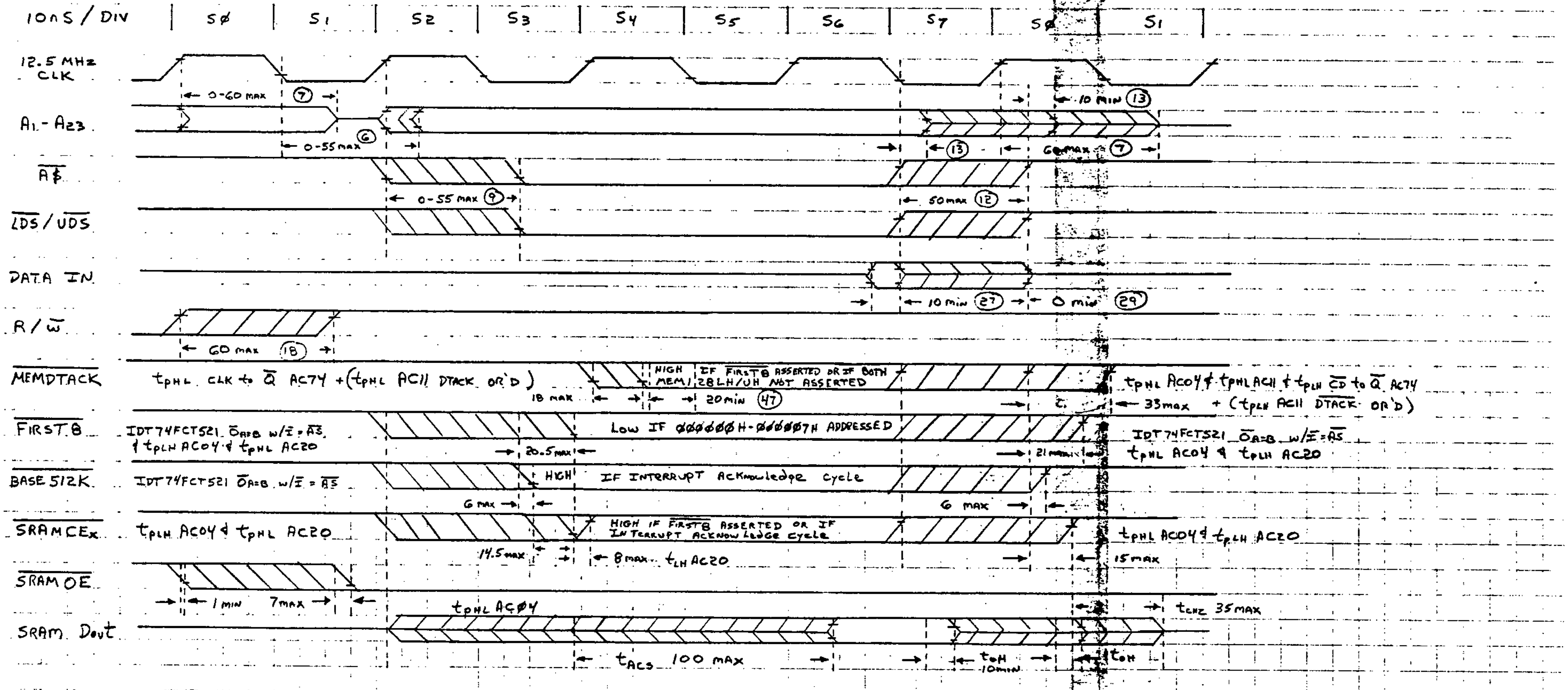
9/7/88 Page 1of 1



SØ
S1
S2
S3
S4
SWAIT
SWAIT
S5
S6
S7
SØ
S1
12.5MHz CLK
A1 - A23
$\overline{AS}$
$\overline{LDS}/\overline{UDS}$
DATA OUT
R/$\overline{W}$
$\overline{MEMDTACK}$
$\overline{FIRST8}$
$\overline{BASE512K}$
$\overline{SRAMCE_x}$
$\overline{SRAM OE}$
$\overline{SRAMWE_{WP}}$
60 MAX (7)
55 MAX (6)
10 min (13)
0-60 MAX (8)(11)
0-55 MAX (9)
50 MAX (12)
0-55 MAX (9)
(55) 10 min
55 MAX (23)
16 min (25)
0-60 MAX (53) (7)
60 MAX (7)
60 MAX (18)
60 MAX (20)
60 MAX (18)
t_{PHL} CLK to $\overline{Q}$, AC74 + (t_{PHL} AC11 $\overline{DTACK}$ OR'D)
18 MAX
HIGH IF $\overline{FIRST8}$ ASSERTED OR IF BOTH MEM128 LH/UH NOT ASSERTED
20 min (47)
t_{PHL} AC04 & t_{PHL} AC11 & t_{PLH} $\overline{CD}$ to $\overline{Q}$ AC74
33 MAX + (t_{PLH} AC11 $\overline{DTACK}$ OR'D)
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$=$\overline{AS}$
& t_{PLH} AC04 & t_{PHL} AC20
20.5 MAX
Low IF ØØØØØØH - ØØØØØ7H ADDRESSED
21 MAX
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$=$\overline{AS}$
t_{PHL} AC04 & t_{PLH} AC20
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$=$\overline{AS}$
6 MAX
HIGH IF INTERRUPT ACKNOWLEDGE cycle
6 MAX
t_{PLH} AC04 & t_{PHL} AC20
14.5 MAX
HIGH IF $\overline{FIRST8}$ ASSERTED OR IF INTERRUPT ACKNOWLEDGE cycle
14.5 MAX
15 MAX
t_{PHL} ACØ4 & t_{PLH} AC20
t_{HL} AC04
7 MAX
7 MAX
7.5 MAX
7.5 MAX
t_{LH} AC04
7 MAX
7 MAX
t_{HL} AC04
t_{LH} AC32
8.5 MAX
t_{HL} AC32
7.5 MAX
8.5 MAX
7.5 MAX
t_{LH} AC32

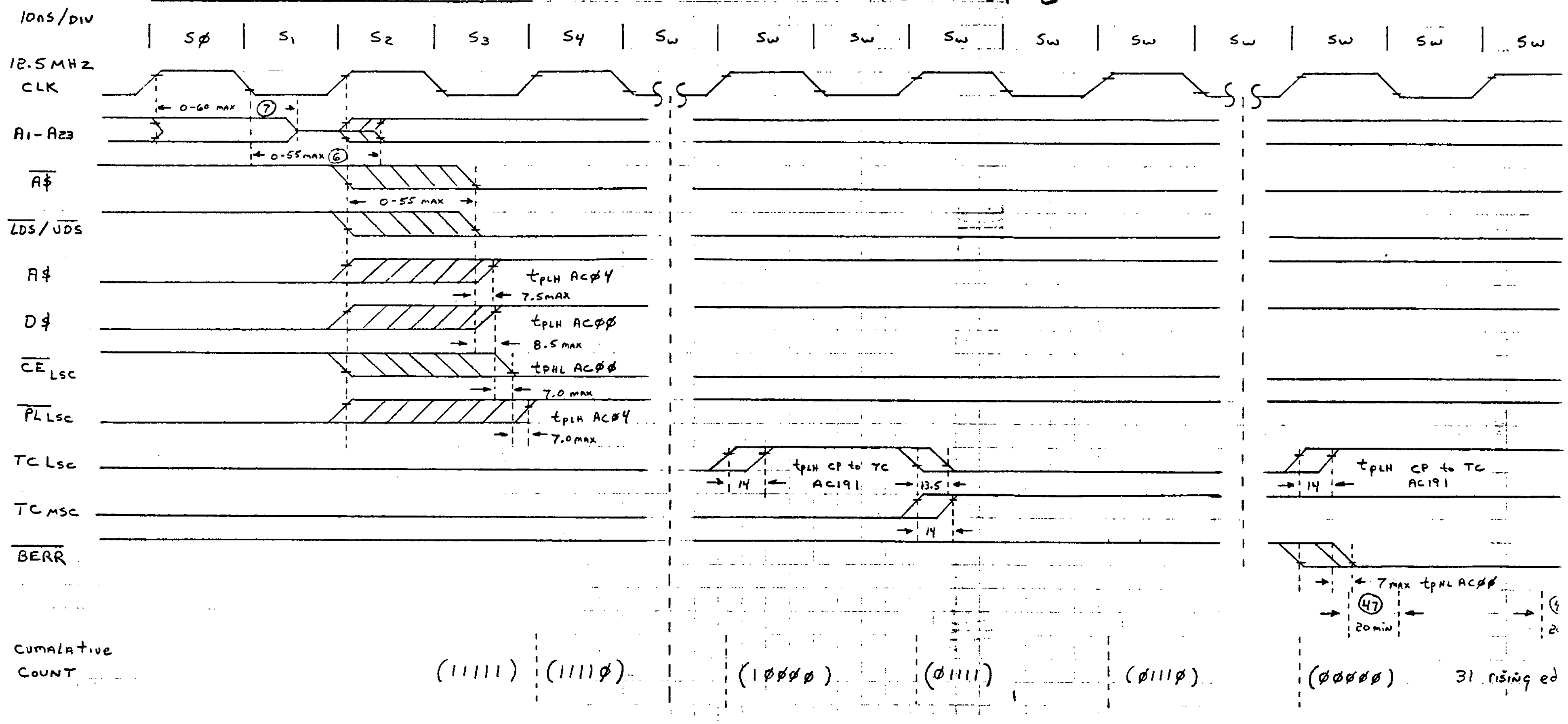
9/7/88 Page 1 of 1

MBA EXO 128K SRAM READ CYCLE

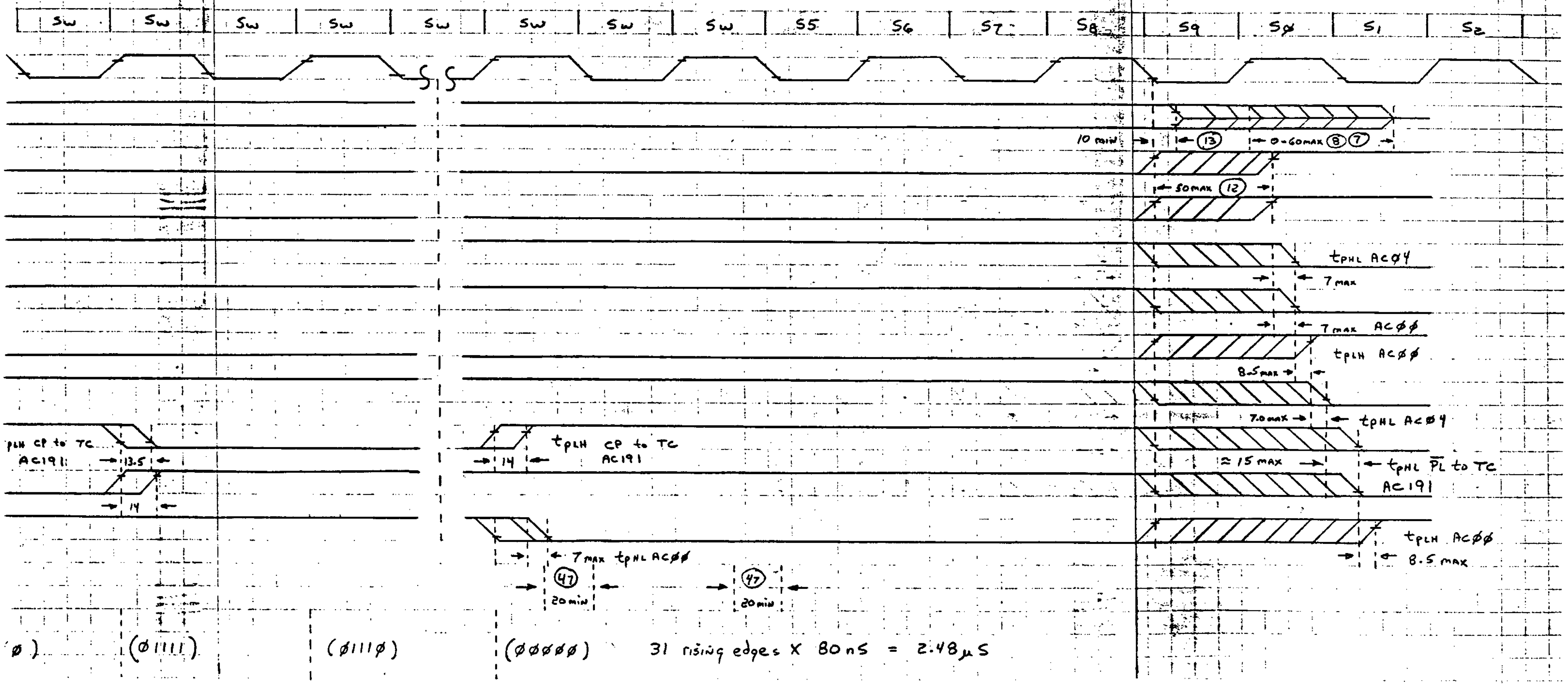


10nS / DIV
S∅
S1
S2
S3
S4
S5
S6
S7
S∅
S1
12.5 MHz CLK
0-60 max
7
10 min 13
A1 - A23
0-55 max 6
13
60 max 7
A$\overline{S}$
0-55 max 9
50 max 12
$\overline{LDS}$ / $\overline{UDS}$
DATA IN
10 min 27
0 min 29
R / $\overline{W}$
60 max 18
$\overline{MEMDTACK}$
t_{PHL} CLK to $\overline{Q}$ AC74 + (t_{PHL} AC11 DTACK OR'D)
18 max
HIGH MEM/
IF $\overline{FIRST8}$ ASSERTED OR IF BOTH 28LH/UH NOT ASSERTED
20 min 47
t_{PHL} AC04 + t_{PHL} AC11 + t_{PLH} $\overline{CD}$ to $\overline{Q}$ AC74
33 max
+ (t_{PLH} AC11 $\overline{DTACK}$ OR'D)
$\overline{FIRST8}$
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$ = $\overline{AS}$ + t_{PLH} AC04 + t_{PHL} AC20
20.5 max
LOW IF ∅∅∅∅∅∅H-∅∅∅∅∅7H ADDRESSED
21 max
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$ = $\overline{AS}$
t_{PHL} AC04 + t_{PLH} AC20
$\overline{BASE512K}$
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$ = $\overline{AS}$
6 max
HIGH
IF INTERRUPT ACKnowledge cycle
6 max
$\overline{SRAMCEx}$
t_{PLH} AC04 + t_{PHL} AC20
14.5 max
HIGH IF $\overline{FIRST8}$ ASSERTED OR IF INTERRUPT ACKNOWLEDGE CYCLE
8 max t_{LH} AC20
t_{PHL} AC04 + t_{PLH} AC20
15 max
$\overline{SRAMOE}$
1 min
7 max
t_{PHL} AC∅4
t_{CHZ} 35 max
SRAM Dout
t_{ACS} 100 max
t_{OH} 10 min
t_{OH}

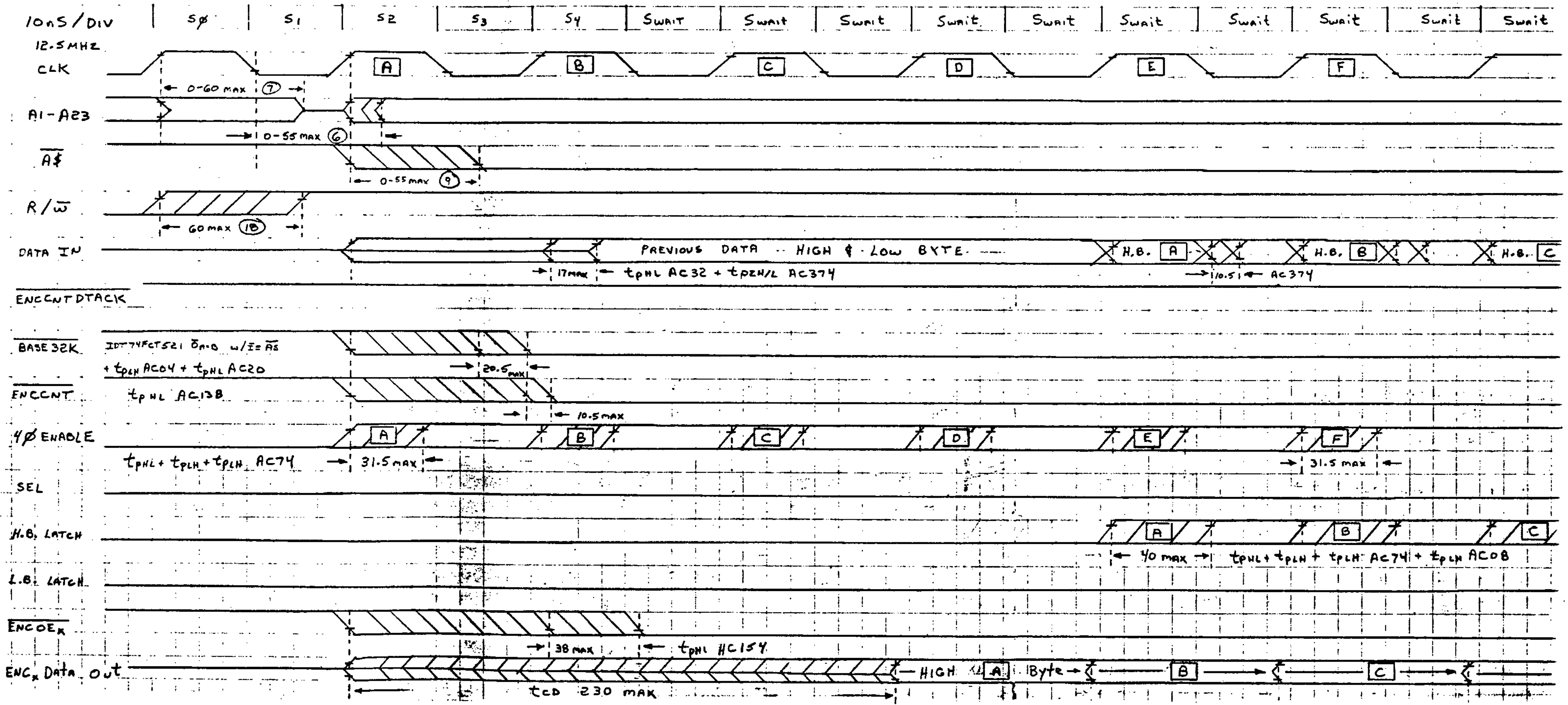
MBA EXO $\overline{\text{BERR}}$ TIMER   5 Oct 88   PAGE 1 of 2



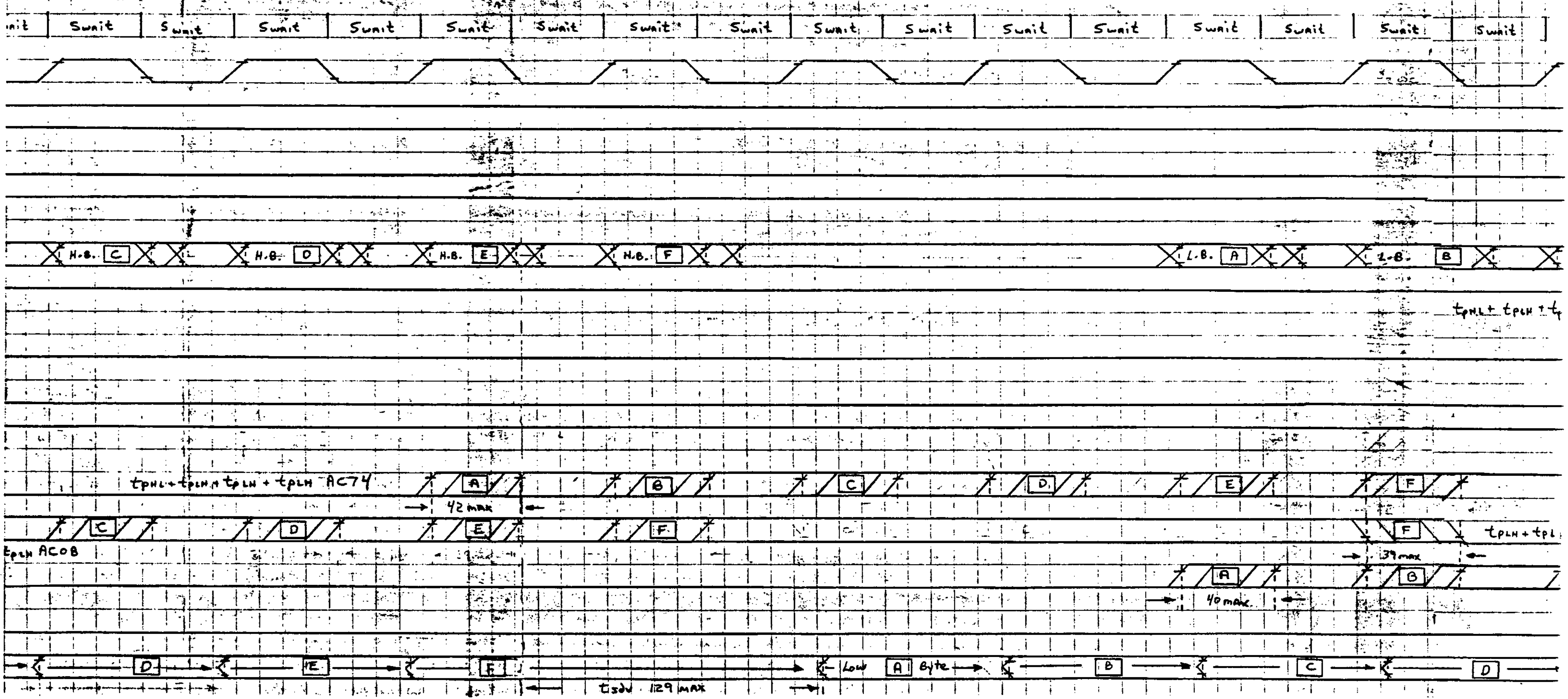
MBA EXO BERR TIMER 5 Oct 88 Page 2 of 2



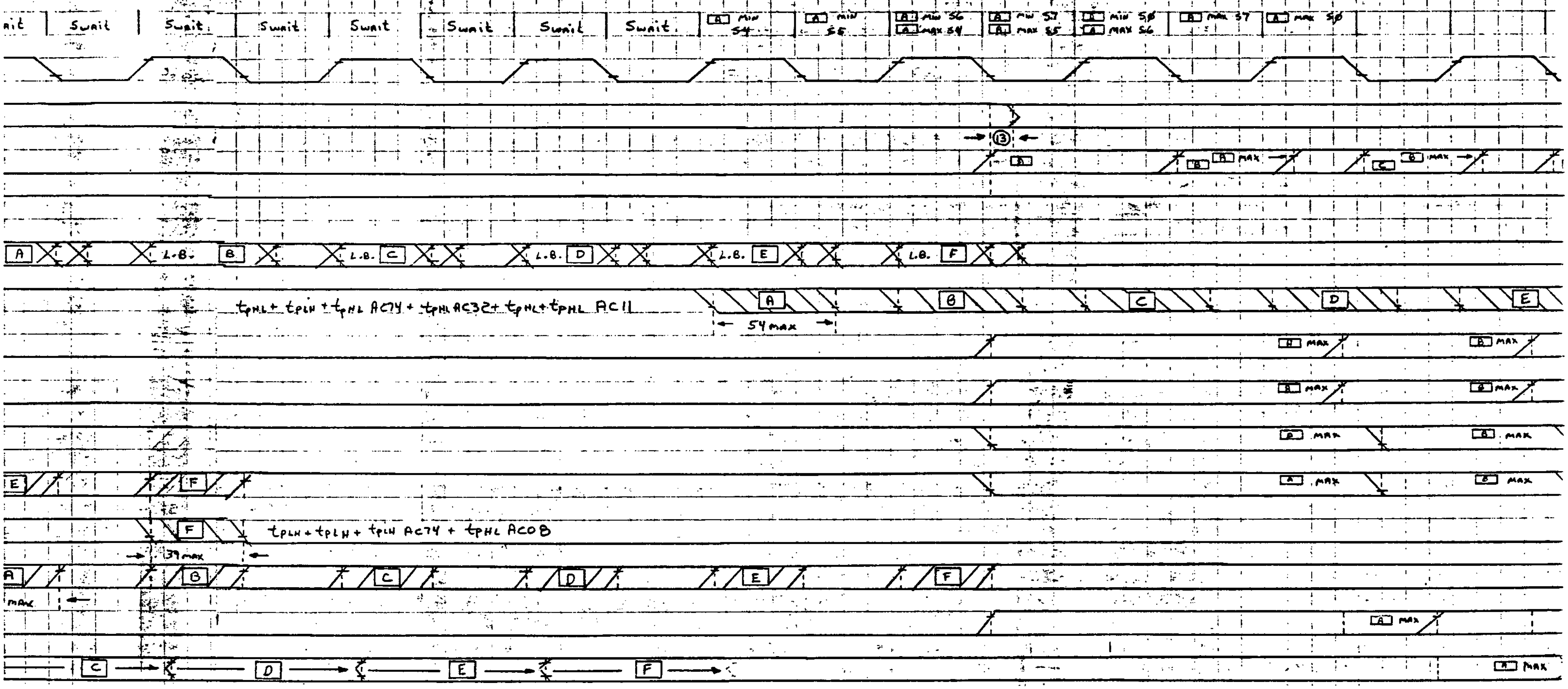
MBA EXOSKELETON ENCODER COUNTER READ CYCLE 9 Sept 88 PAGE 1 of 4



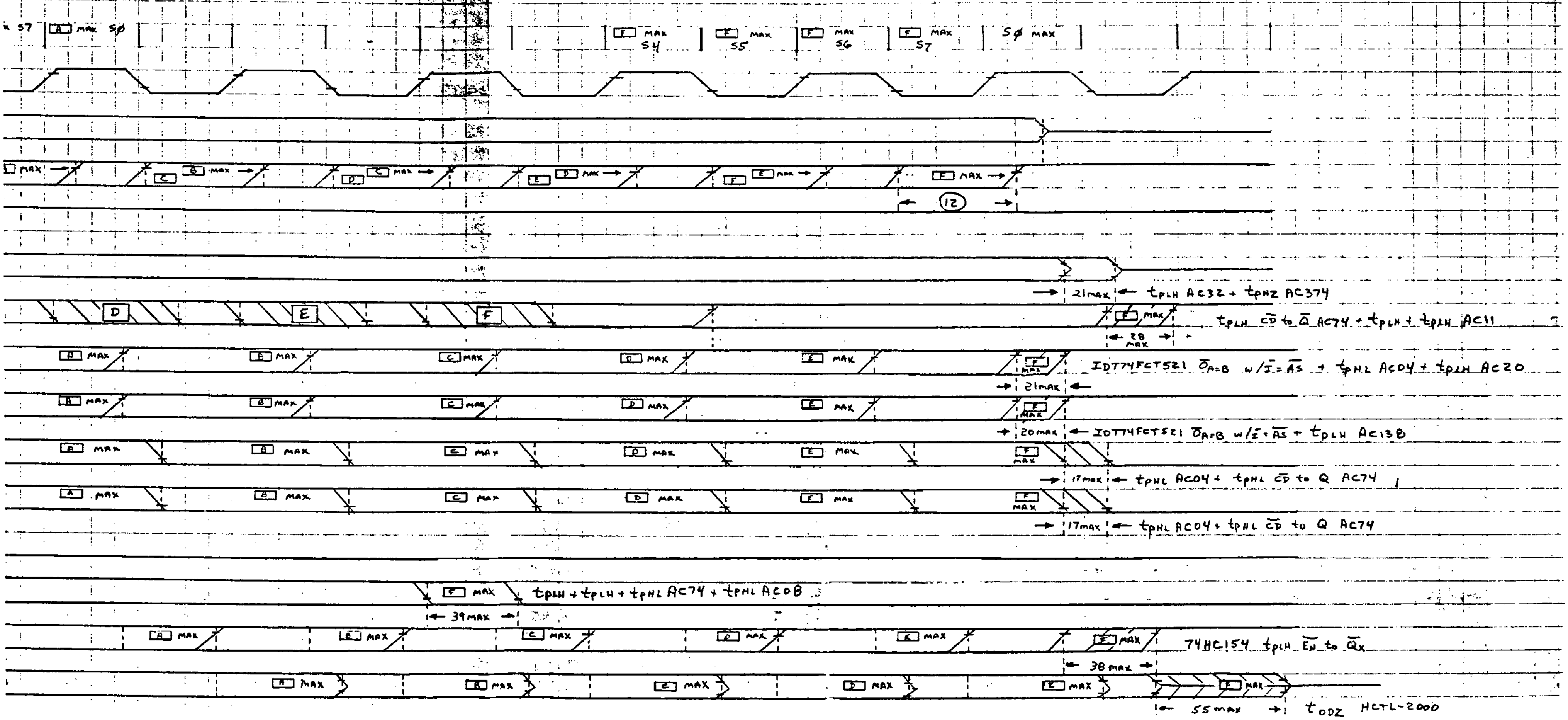
MBA EXOSKELETON ENCODER COUNTER READ CYCLE 9 Sept 88 PAGE 2 of 4



MBA EXOSKELETON ENCODER Counter READ Cycle 9 Sept 88 Page 3 of 4



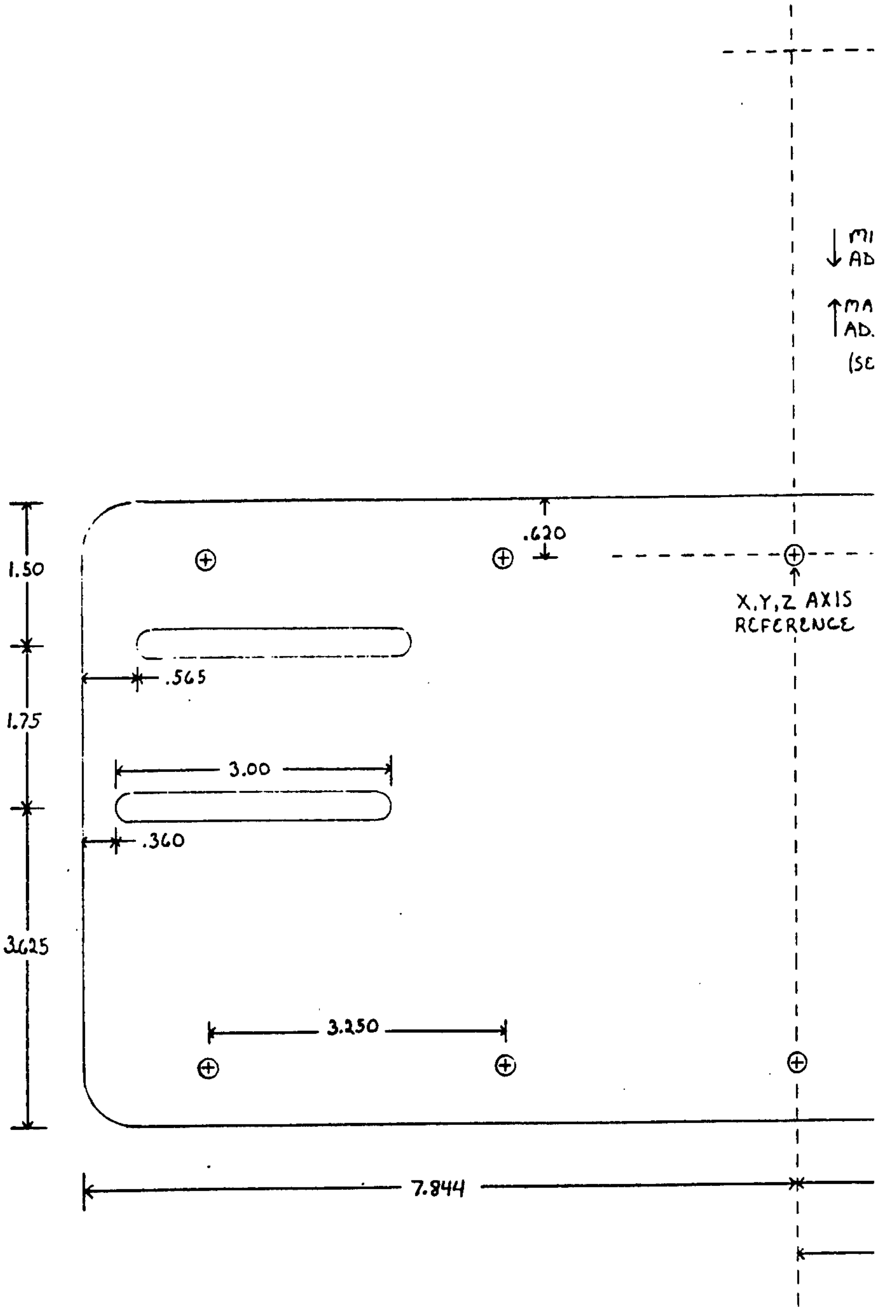
MBA EXOSKELETON ENCODER COUNTER READ Cycle 9 Sept 88 PAGE 4 of 4

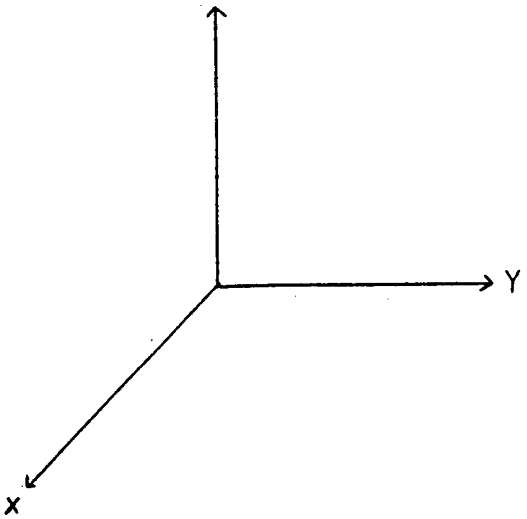

S7
A MAX S0
F MAX S4
F MAX S5
F MAX S6
F MAX S7
S0 MAX
B MAX
C
C MAX
D
D MAX
E
E MAX
F
F MAX
12
21max
t_{PLH} AC32 + t_{PHZ} AC374
D
E
F
t_{PLH} $\overline{CD}$ to $\overline{Q}$ AC74 + t_{PLH} + t_{PLH} AC11
28 MAX
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$ = $\overline{AS}$ + t_{PHL} AC04 + t_{PLH} AC20
21MAX
20max
IDT74FCT521 $\overline{O}_{A=B}$ w/$\overline{I}$ = $\overline{AS}$ + t_{PLH} AC138
A MAX
17max
t_{PHL} AC04 + t_{PHL} $\overline{CD}$ to Q AC74
17max
t_{PHL} AC04 + t_{PHL} $\overline{CD}$ to Q AC74
F MAX
t_{PLH} + t_{PLH} + t_{PHL} AC74 + t_{PHL} AC08
39max
74HC154 t_{PLH} $\overline{E_N}$ to $\overline{Q_x}$
38 max
55max
t_{ODZ} HCTL-2000

9.0 Appendix A: MBAssociates Backpack

9.3 Mechanical Drawings

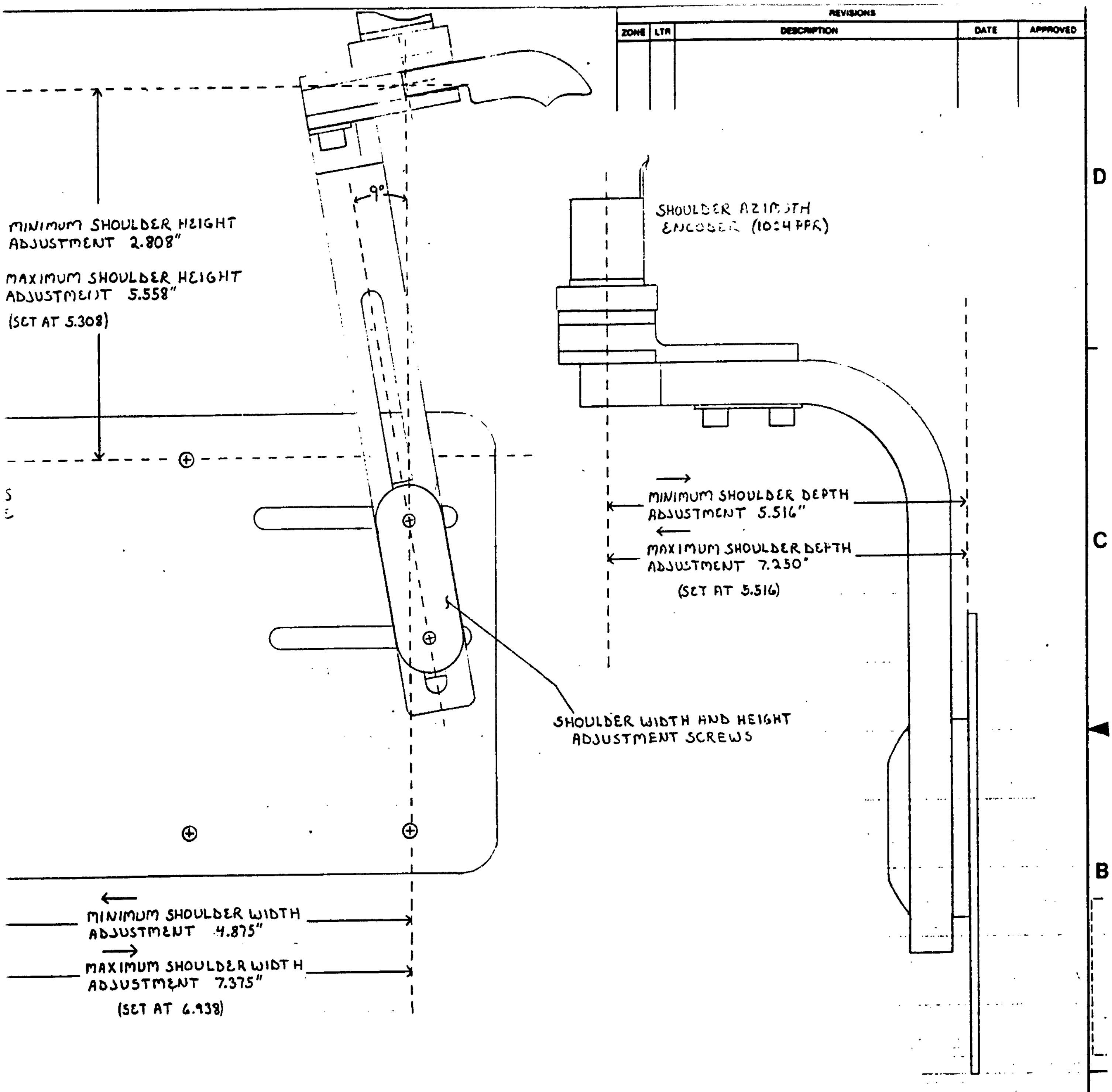
This page intentionally left blank.


MI
AD
MA
AD.
(SE
.620
1.50
X,Y,Z AXIS
REFERENCE
.565
1.75
3.00
.360
3.625
3.250
7.844


Z
Y
X

REVISIONS

ZONE	LTR	DESCRIPTION	DATE	APPROVED



9°
MINIMUM SHOULDER HEIGHT ADJUSTMENT 2.808"
MAXIMUM SHOULDER HEIGHT ADJUSTMENT 5.558"
(SET AT 5.308)
SHOULDER AZIMUTH ENCODER (1024 PPR)
MINIMUM SHOULDER DEPTH ADJUSTMENT 5.516"
MAXIMUM SHOULDER DEPTH ADJUSTMENT 7.250"
(SET AT 5.516)
SHOULDER WIDTH AND HEIGHT ADJUSTMENT SCREWS
MINIMUM SHOULDER WIDTH ADJUSTMENT 4.875"
MAXIMUM SHOULDER WIDTH ADJUSTMENT 7.375"
(SET AT 6.938)
D
C
B
A

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

QTY REQD PER ASSY

PARTS LIST

PART NO. | N/A | F/A | NEXT ASSY | USED ON

QTY REQD | APPLICATION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL

FINISH

CONTRACT NO.		
DRAWN BY	J. Logan	DATE 1-30-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

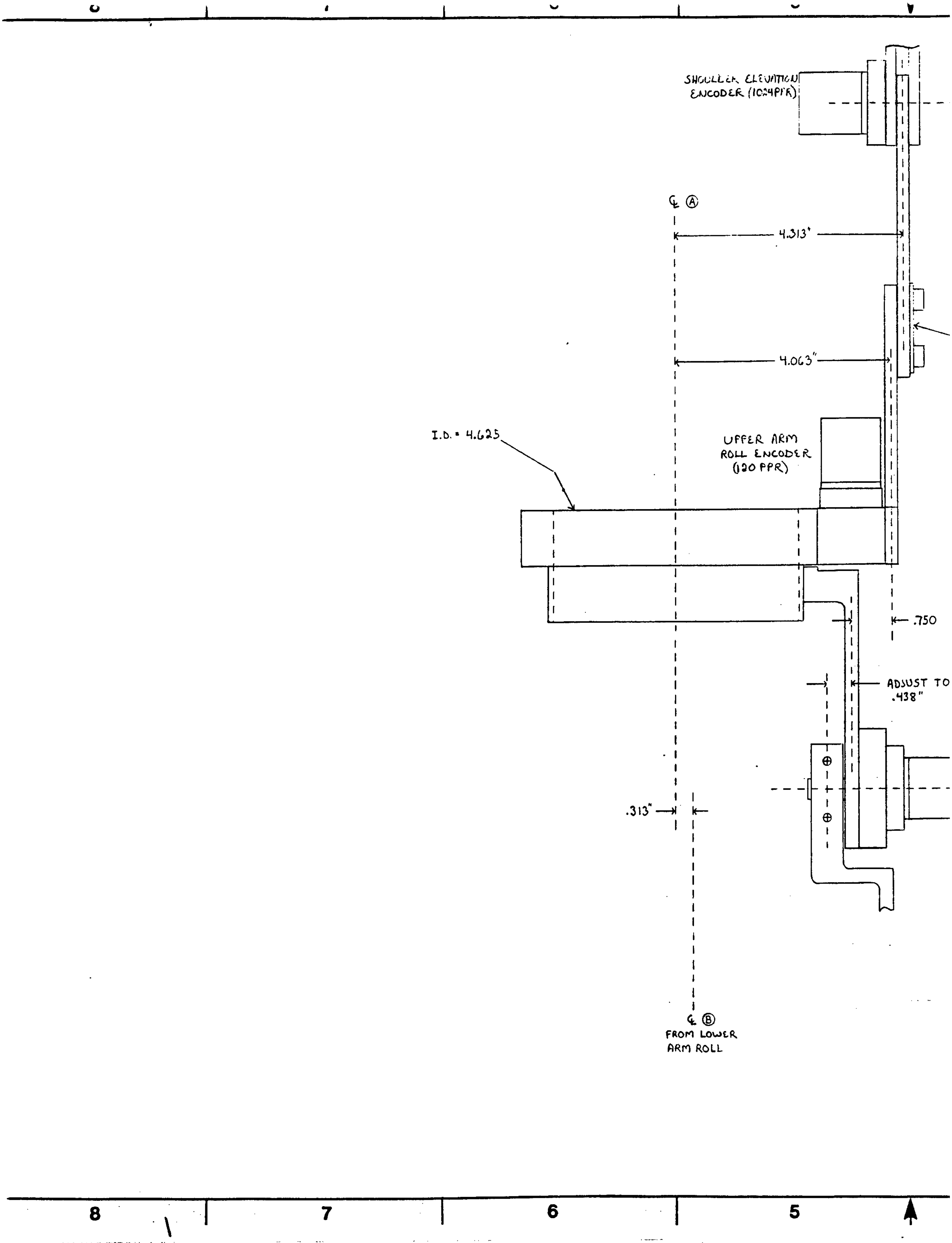
SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELETON
MOUNTING PLATE TO SHOULDER AZIMUTH
ENCODER/JOINT POSITIONS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D	14590		

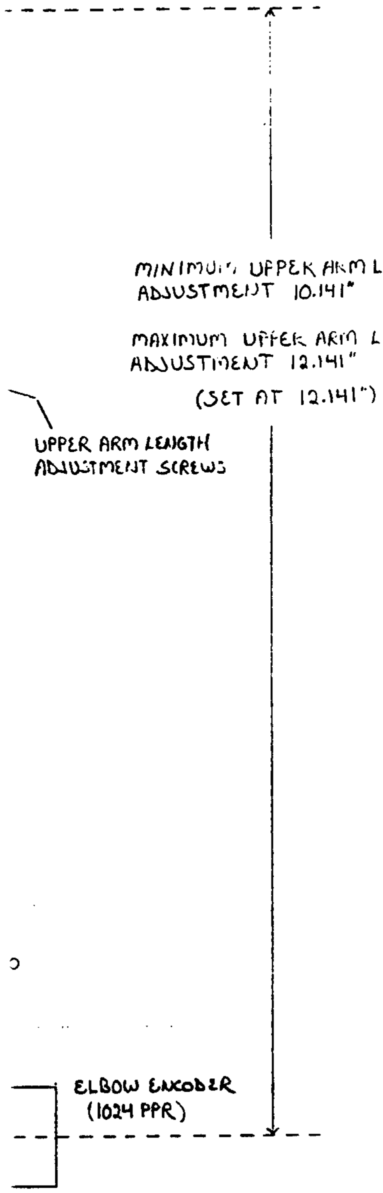
SCALE | RELEASE DATE | SHEET 2 OF 6

4 3 2 1


SHOULDER ELEVATION ENCODER (1024PPR)
℄ Ⓐ
4.313"
4.063"
I.D. = 4.625
UPPER ARM ROLL ENCODER (120 PPR)
.750
ADJUST TO .438"
.313"
℄ Ⓑ FROM LOWER ARM ROLL
8
7
6
5

REVISIONS

ZONE	LTR	DESCRIPTION	DATE	APPROVED


MINIMUM UPPER ARM LENGTH ADJUSTMENT 10.141"
MAXIMUM UPPER ARM LENGTH ADJUSTMENT 12.141"
(SET AT 12.141")
UPPER ARM LENGTH ADJUSTMENT SCREWS
ELBOW ENCODER (1024 PPR)

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO

QTY REQD PER ASSY

PARTS LIST

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

XX = ANGLES = ± 30'
XXX = FRACTIONS = ± 1/32
XXXX = BASIC

ALL SURFACES ✓

MATERIAL

FINISH

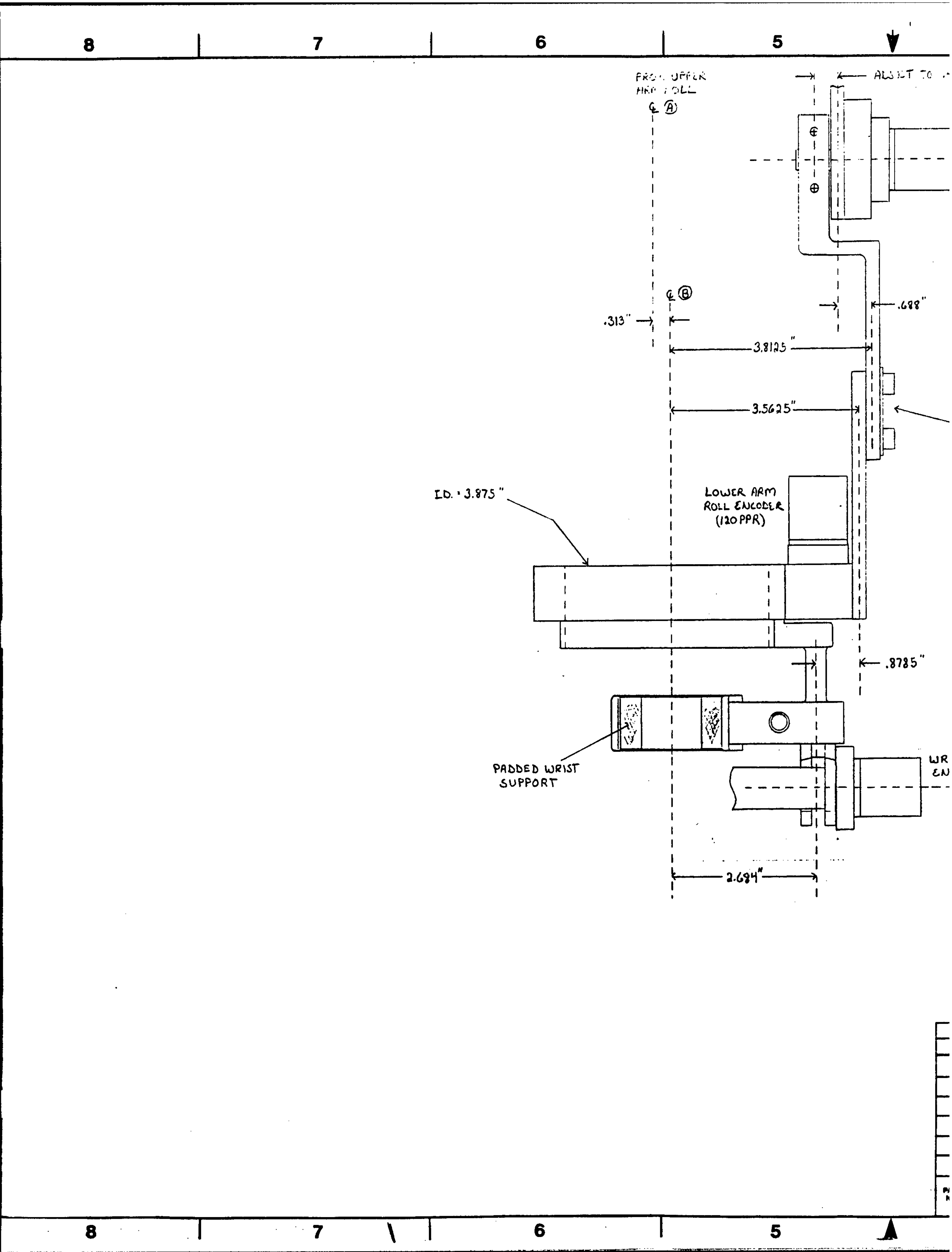
		DATE
CONTRACT NO.		
DRAWN BY	J Logan	1-31-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

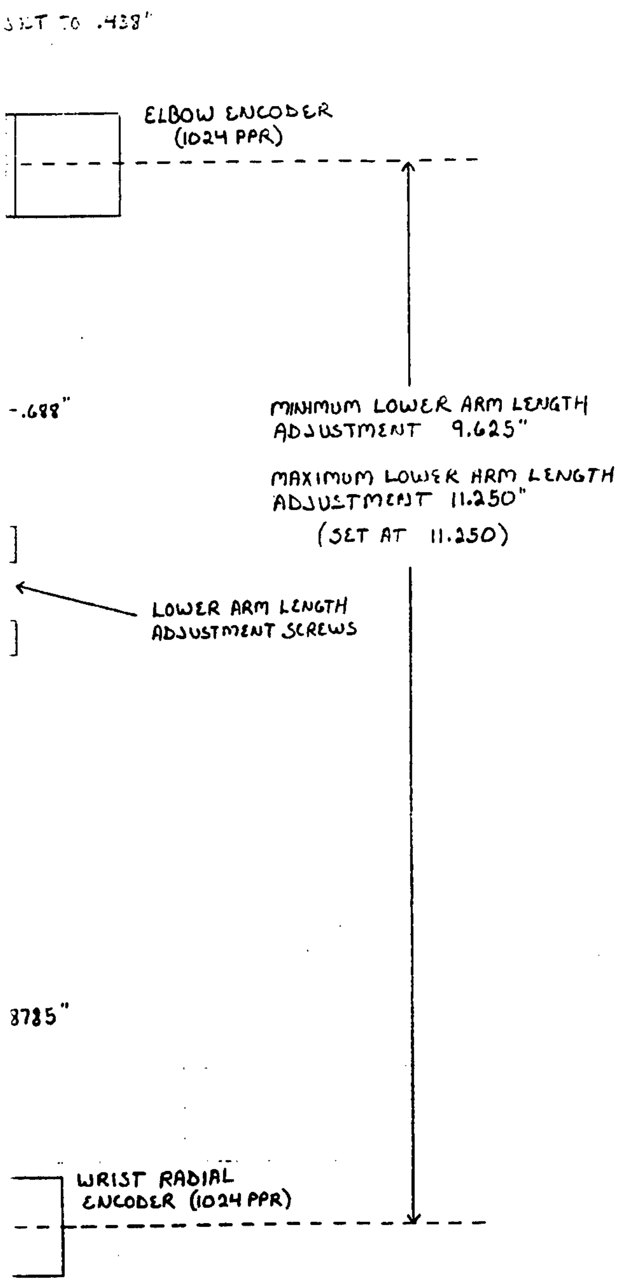
MBA EXOSKELTON
SHOULDER ELEVATION, UPPER ARM ROLL, ELBOW ENCODER/JOINT POSITIONS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D	14590		

SCALE RELEASE DATE SHEET 3 OF 6


8
7
6
5
FROM UPPER
ARM ROLL
℄ (A)
ADJUST TO
℄ (B)
.313"
.688"
3.8125"
3.5625"
I.D. = 3.875"
LOWER ARM
ROLL ENCODER
(120 PPR)
.8785"
PADDED WRIST
SUPPORT
WR
EN
2.684"
8
7
6
5

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED


SET TO .438"
ELBOW ENCODER (1024 PPR)
-.688"
MINIMUM LOWER ARM LENGTH ADJUSTMENT 9.625"
MAXIMUM LOWER ARM LENGTH ADJUSTMENT 11.250"
(SET AT 11.250)
LOWER ARM LENGTH ADJUSTMENT SCREWS
3785"
WRIST RADIAL ENCODER (1024 PPR)

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

QTY REQD PER ASSY

PART NO.	N/A	P/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL

FINISH

CONTRACT NO.

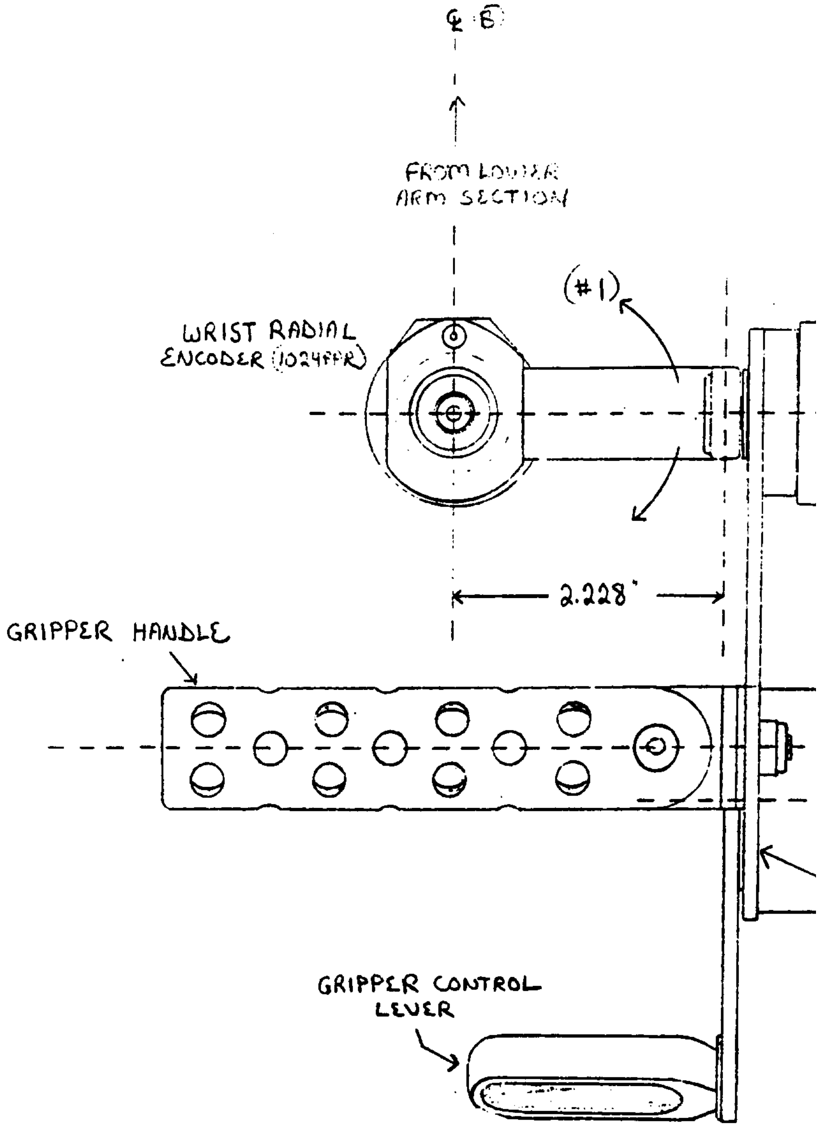
		DATE
DRAWN BY	J Logan	1-31-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELTON
ELBOW, LOWER ARM ROLL, WRIST RADIAL
ENCODER/JOINT POSITIONS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D	14590		

SCALE | RELEASE DATE | SHEET 4 OF 6



QTY REQD PER ASSY							

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

PART OR IDENTIFYING NO. | CODE IDENT

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX =
.XXX =
.XXXX = BASIC

ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL

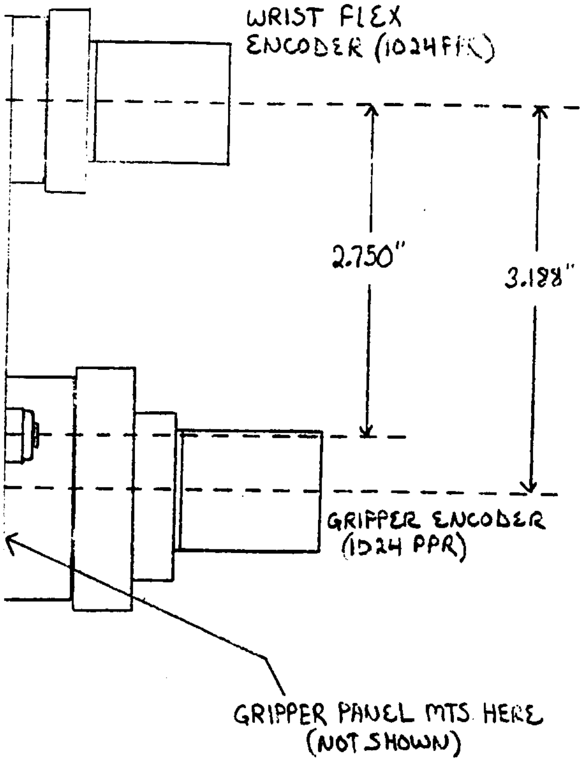
FINISH

CONTR
DRAWN
CHECK
DESIGN
PROJEC
CUSTOM
QUALIT ASSUR
MANUF

MRC

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

D



C

B

NOTE (#1) WRIST RADIAL HAS ≈ 35° OF FREE MOTION
IN DIRECTIONS SHOWN

ENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

CONTRACT NO.		
DRAWN BY	JLogan	DATE 1-31-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELETON (FRONT VIEW)
WRIST RADIAL, WRIST FLEX, AND GRIPPER
ENCODER / JOINT POSITIONS

A

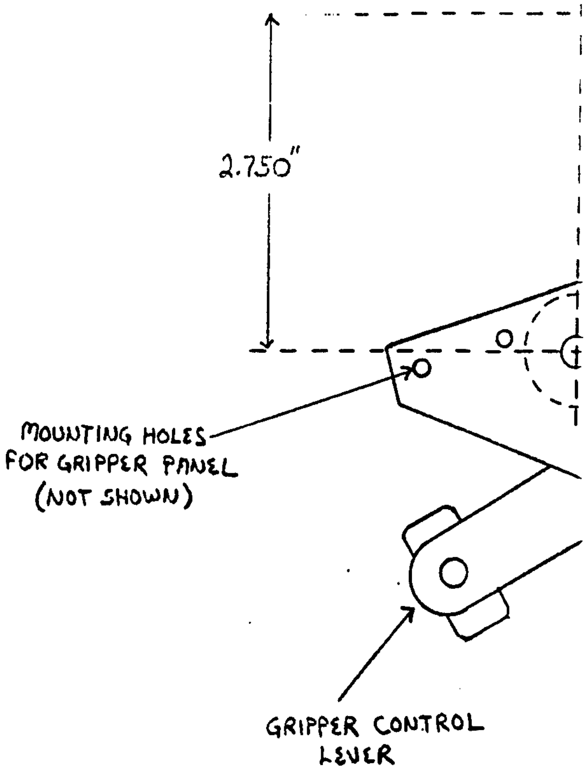
SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE	RELEASE DATE	SHEET 5 OF 6

4 3

D

C



B

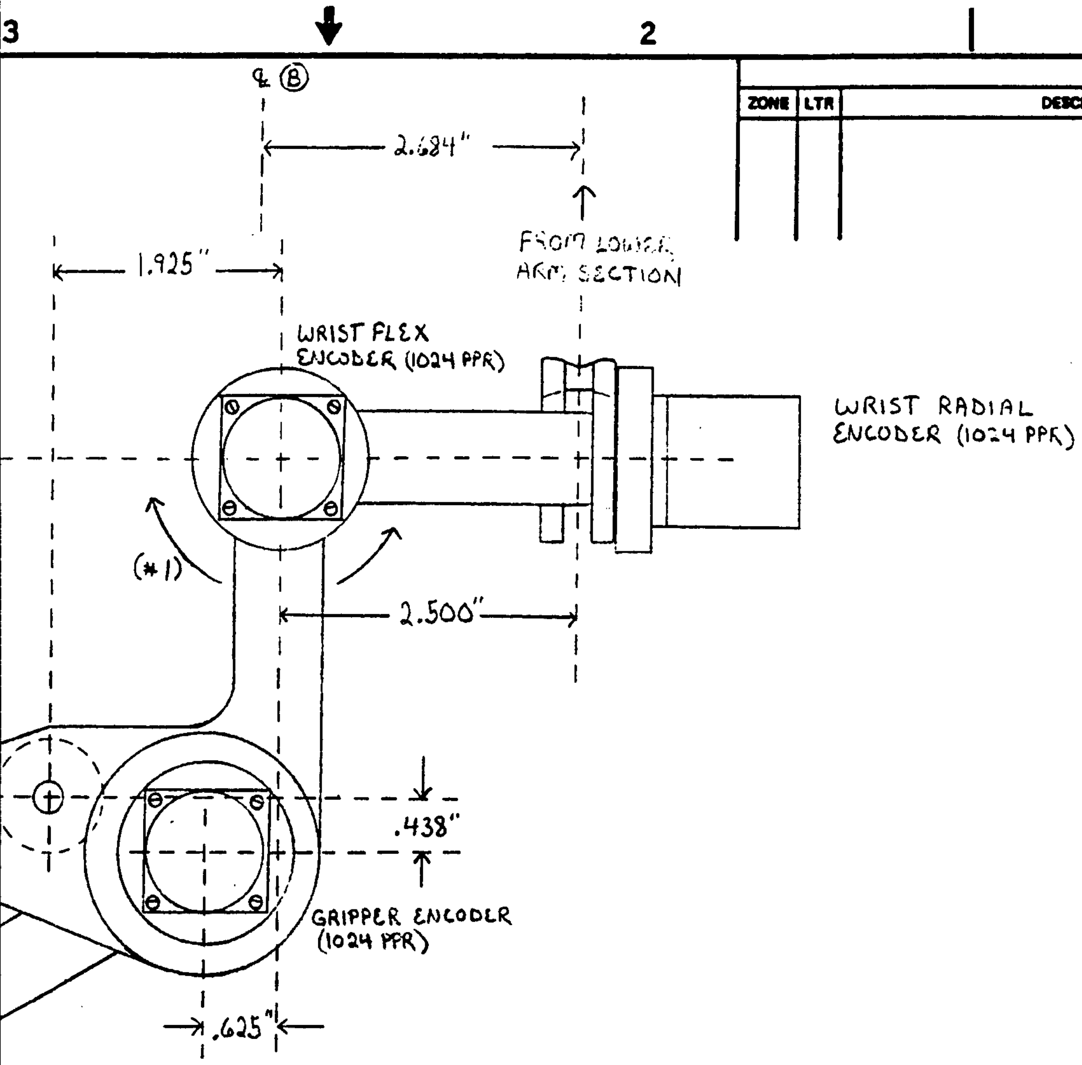
A

QTY REQD PER ASSY				
PART NO.	N/A	F/A	NEXT ASSY	U
	QTY REQD		APPLICATION	

MRC

4 3

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED


℄ Ⓑ
2.684"
1.925"
FROM LOWER ARM SECTION
WRIST FLEX ENCODER (1024 PPR)
WRIST RADIAL ENCODER (1024 PPR)
(#1)
2.500"
.438"
GRIPPER ENCODER (1024 PPR)
.625"

NOTE - (#1) WRIST FLEX HAS ≈ 90° OF FREE MOTION
IN DIRECTIONS SHOWN

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.
PARTS LIST						

USED ON

CATION

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:
.XX = ±.01 ANGLES = ± 30'
.XXX = ±.005 FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL

FINISH

CONTRACT NO.		
DRAWN BY	J. Logan	DATE 1-31-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

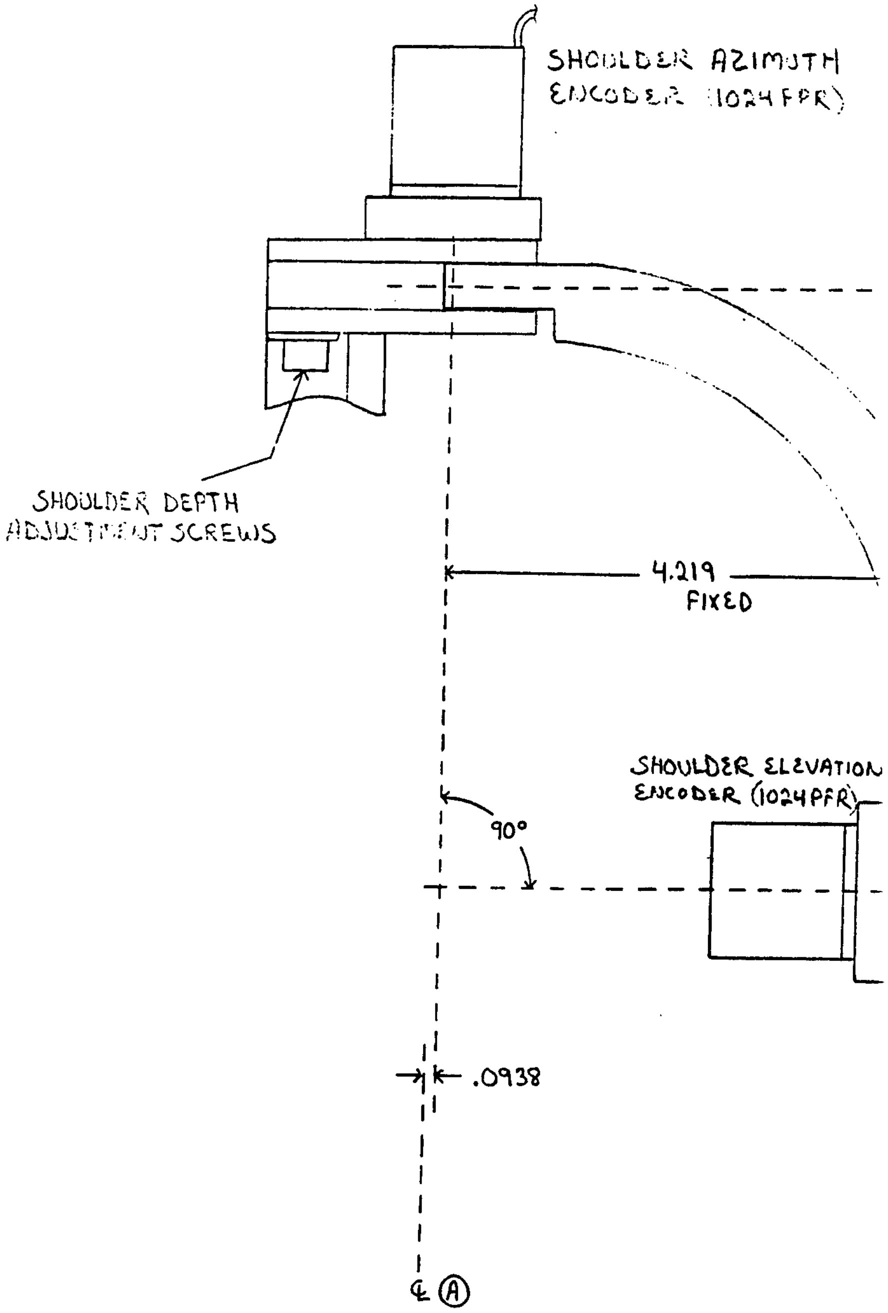
MBA EXOSKELETON (SIDE VIEW)
WRIST RADIAL, WRIST FLEX, AND GRIPPER
ENCODER / JOINT POSITIONS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE	RELEASE DATE	SHEET 6 OF 6

4 3

D


SHOULDER AZIMUTH ENCODER (1024 FPR)
SHOULDER DEPTH ADJUSTMENT SCREWS
4.219 FIXED
SHOULDER ELEVATION ENCODER (1024PFR)
90°
.0938
℄ (A)

C

B

A

								PART OR IDENTIFYING NO.	CODE IDENT	NC
QTY REQD PER ASSY										

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX =
.XXX =
.XXXX = BASIC

ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL

FINISH

CONTRACT

DRAWN BY

CHECKED

DESIGN

PROJECT

CUSTOMER

QUALITY ASSURANC

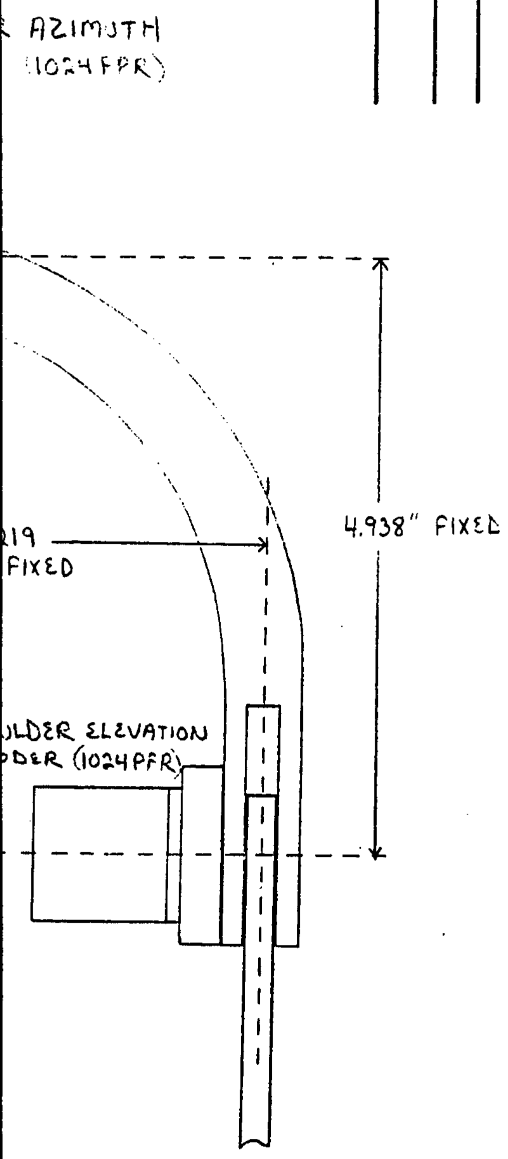
MANUFACTU

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

MRC 056203

4 3

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED



NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

SPECIFIED
N INCHES
ES:
NGLES = ± 30'
TIONS = ± 1/32
ES ✓

CONTRACT NO.		
DRAWN BY	J Logan	DATE 1-31-90
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

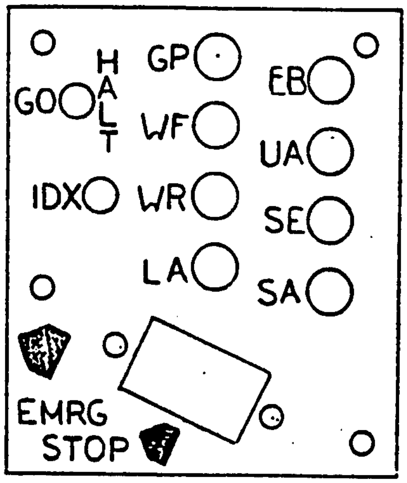
SRL
SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSELETON
SHOULDER AZIMUTH TO SHOULDER ELEVATION
ENCODER/JOINT POSITIONS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE	RELEASE DATE	SHEET 2 OF 6

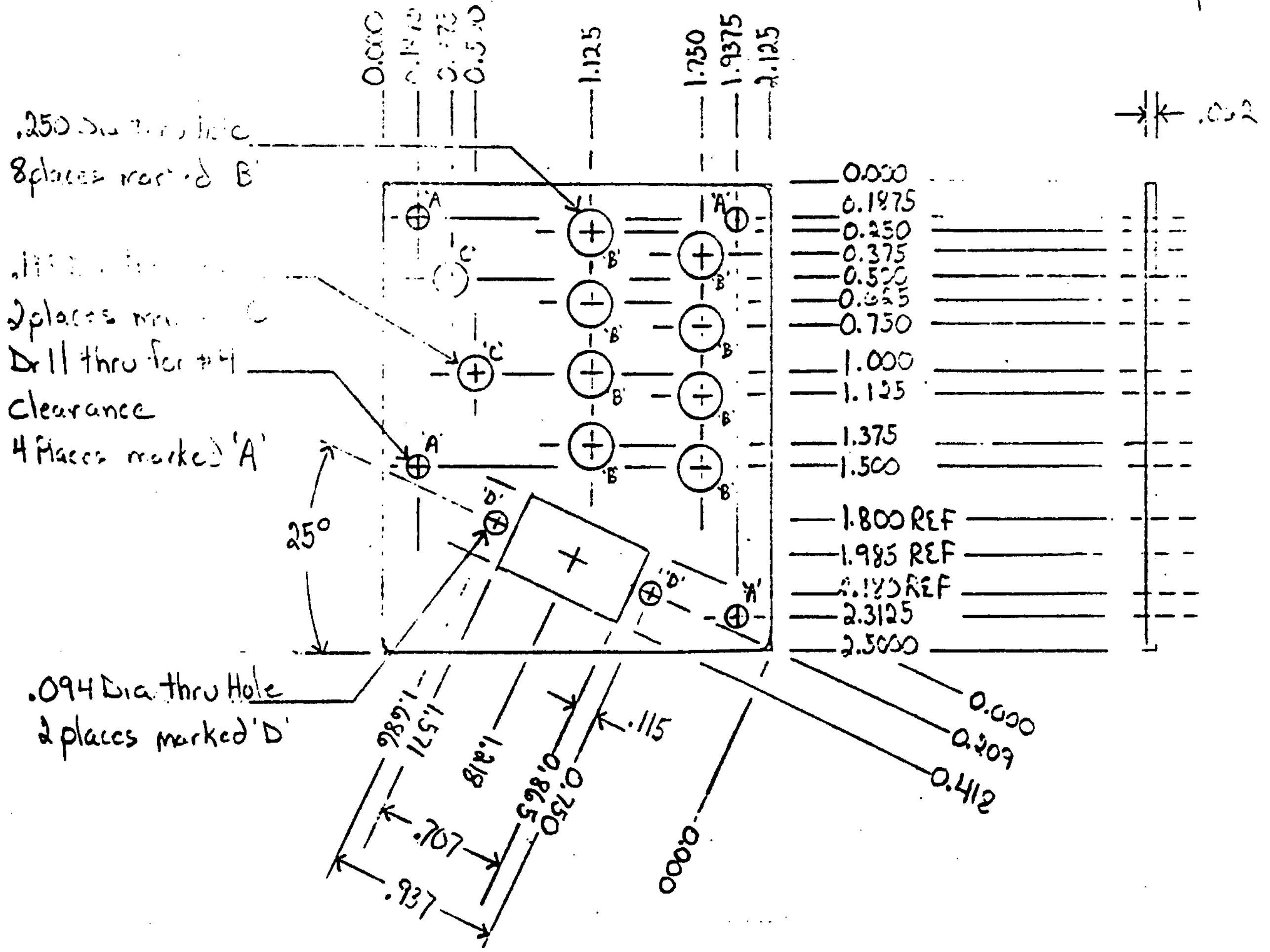
2



NOTE: Left Panel Shown, Right Panel
Is Mirror Image

MBA EXOSKELETON
GRIPPER PANEL LABELING

RT + LT Go
x Rt part shown

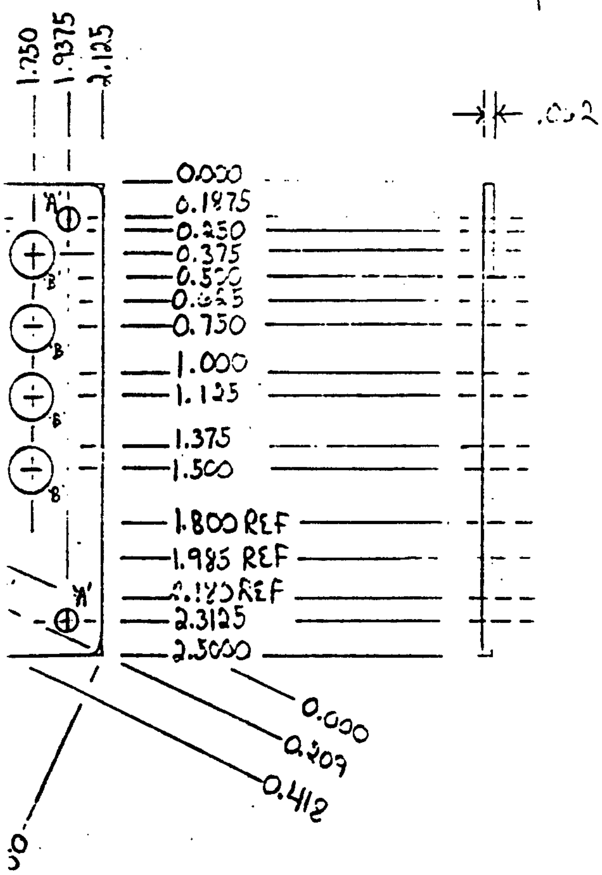


material: .062 in Allumi

MBA E

GRIPPER

RT + LT Configurations
* (Rt panel shown, Lt panel is Mirror image)

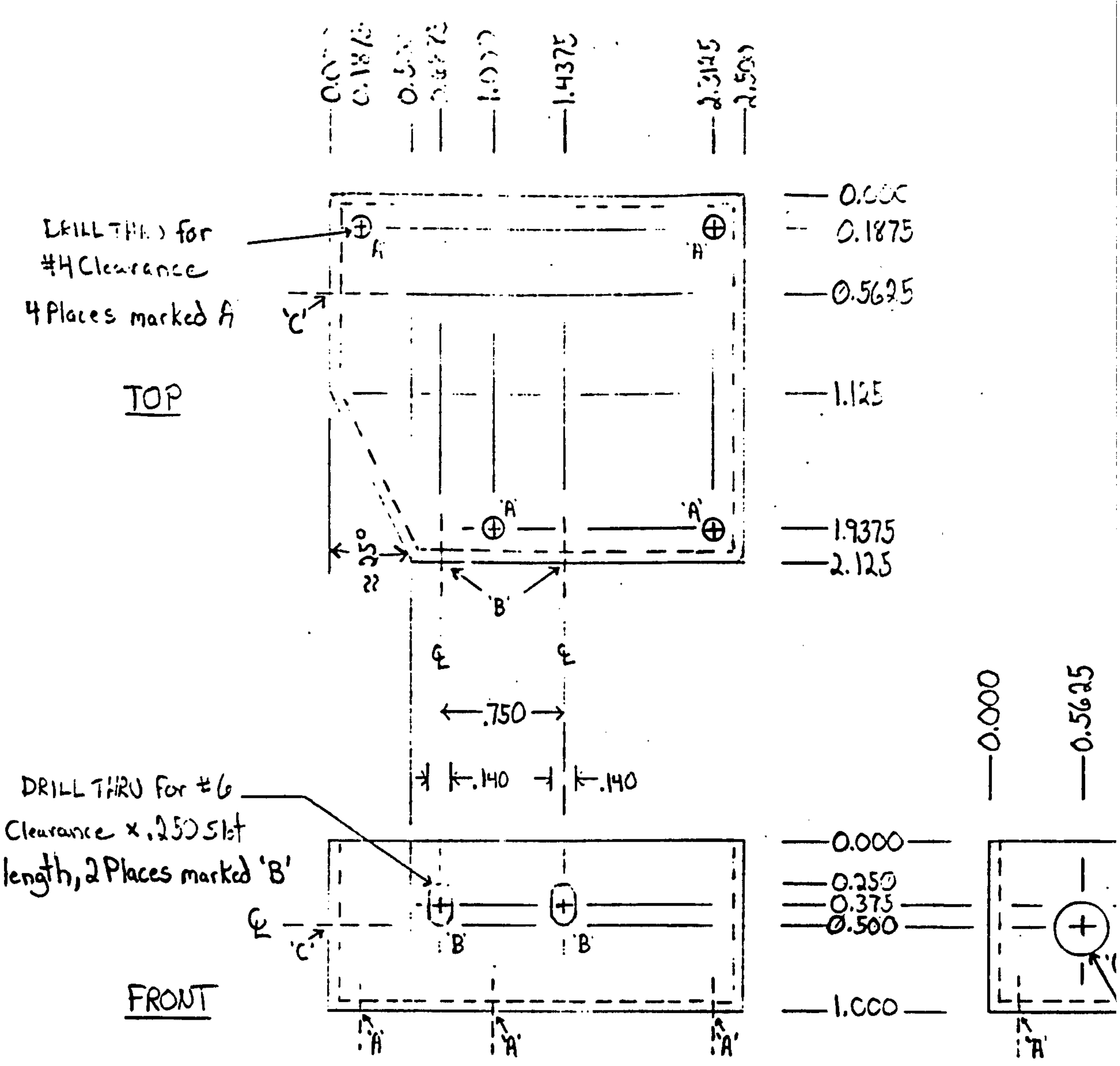


material: .062 in Alluminum

MBA EXOSKELETON

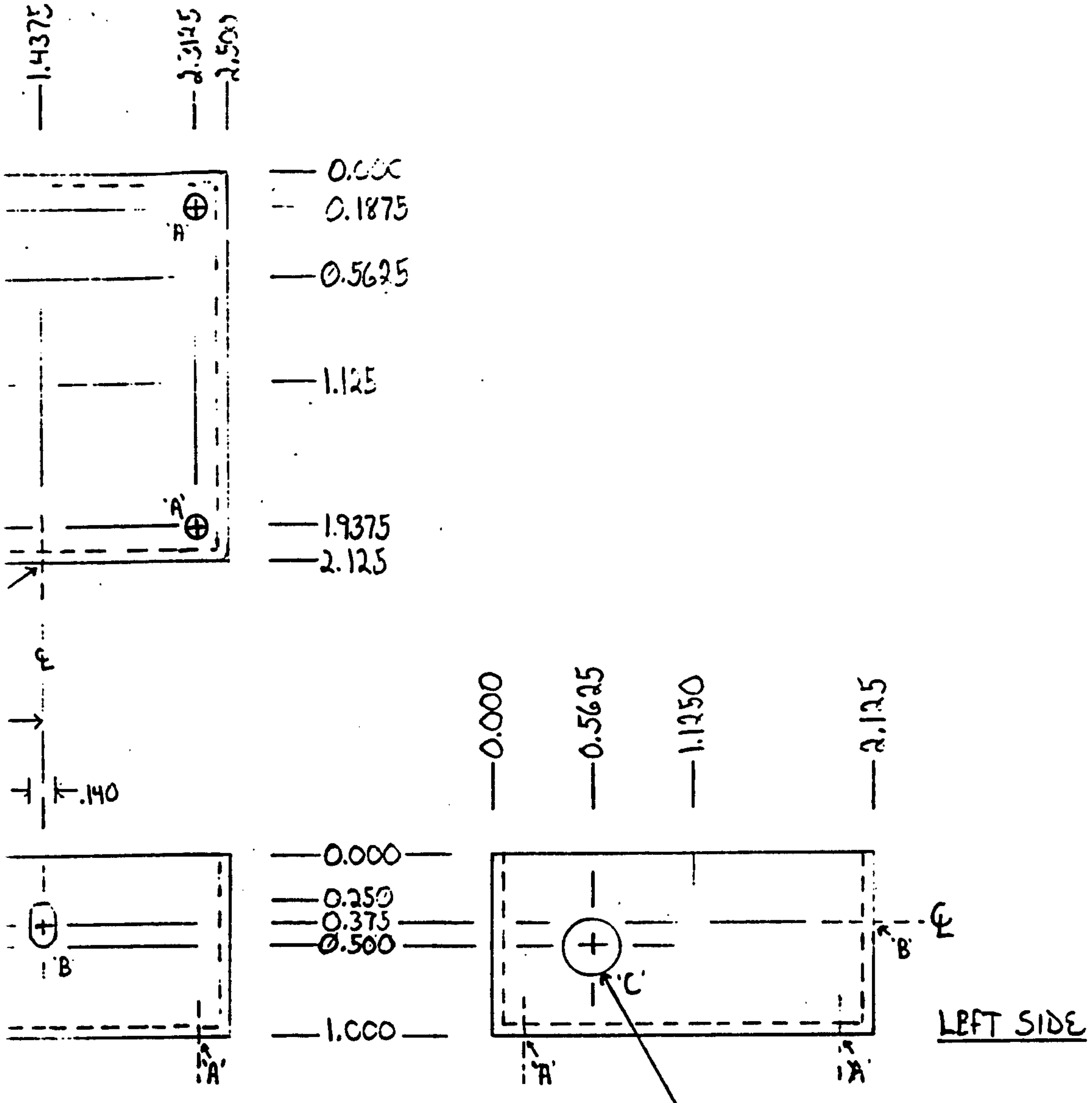
GRIPPER PANEL COVER

2



MATERIAL: .062" Aluminum

ME
GR
(L

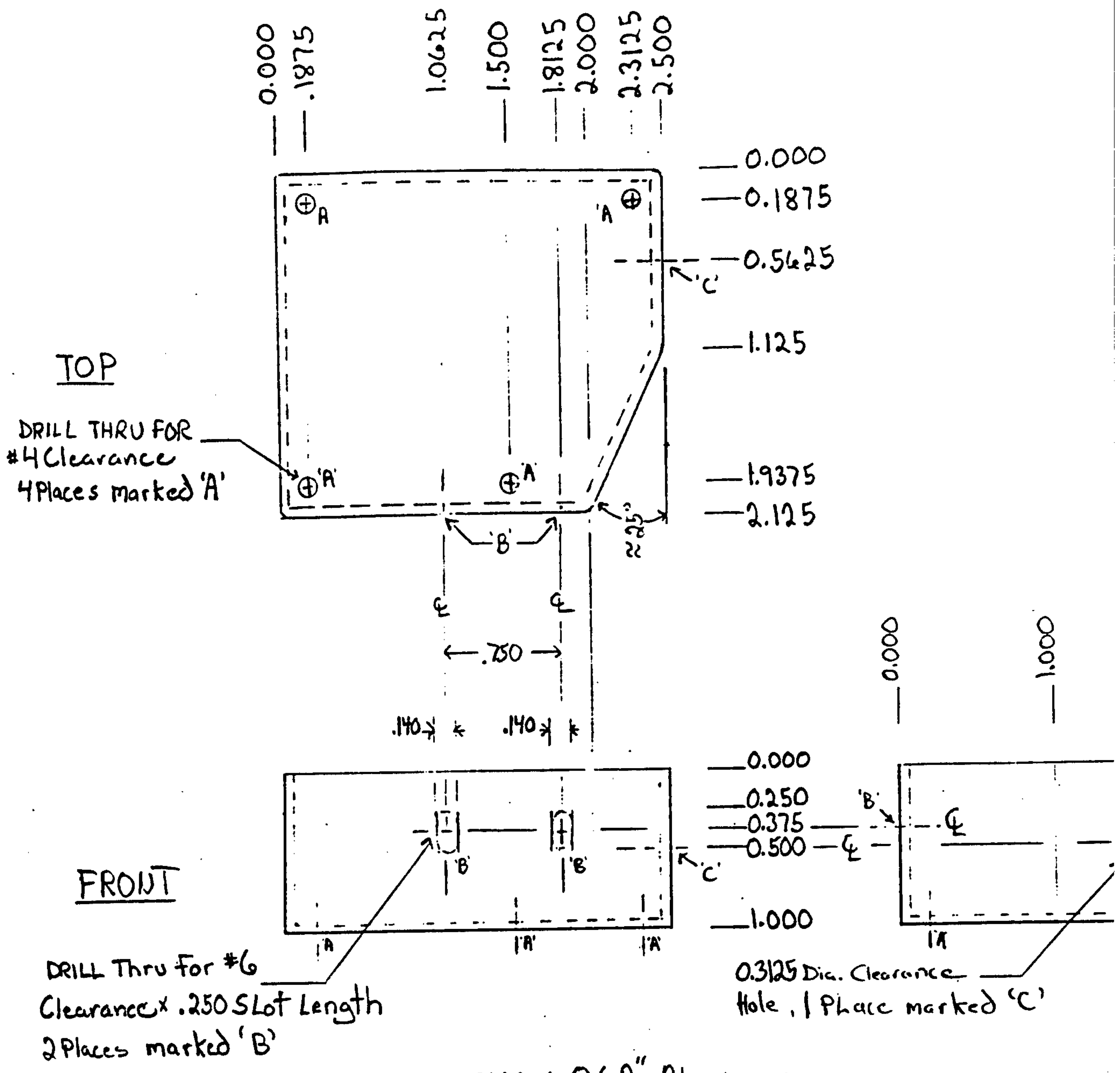


0.3125 Dia Clearance
Hole, 1 Place marked 'C'

62" Aluminum

MBA EXOSKELETON
GRIPPER PANEL ENCLOSURE
(LEFT)

2



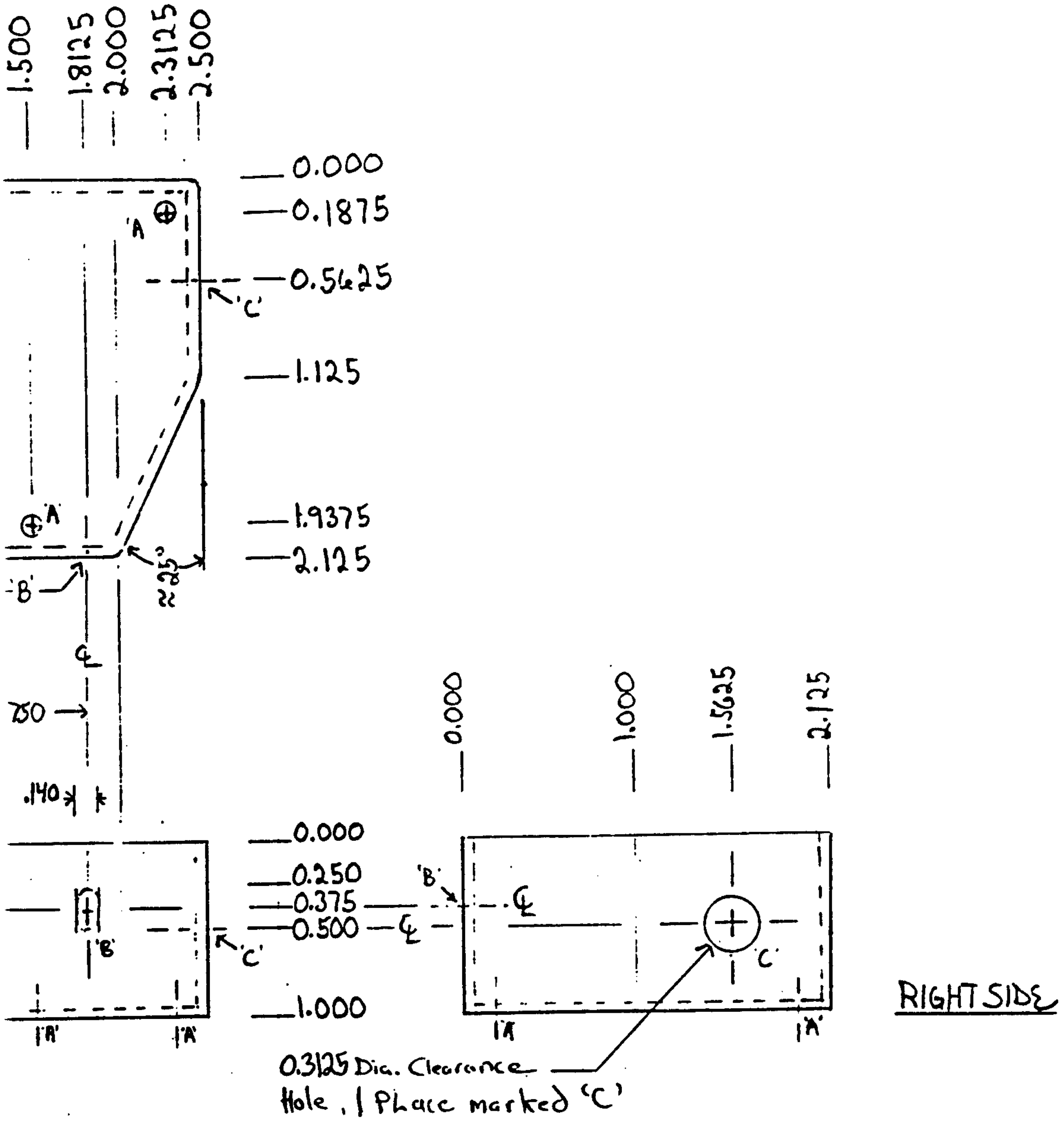
DRILL THRU FOR #4 Clearance 4 Places marked 'A'

DRILL Thru For #6 Clearance x .250 SLot Length 2 Places marked 'B'

0.3125 Dia. Clearance Hole, 1 PLace marked 'C'

MATERIAL : .062" Aluminum

MBA EX
GRIPPER
(Right)



1AL : .062" Aluminum

MBA EXOSKELETON
GRIPPER PANEL ENCLOSURE
(Right)

2

4 3

D

C

B

A

MRC

QTY REQD PER ASSY							

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

PART OR IDENTIFYING NO. | CODE IDENT

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL
303 Stainless

FINISH

CONTRA
DRAWN
CHECKE
DESIGN
PROJEC
CUSTOM
QUALITY ASSURA
MANUFA

4 3

2 1

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

A

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL
303 Stainless

FINISH

USED ON

ATION

CONTRACT NO.

DRAWN BY	J Logan	DATE 12-28-87
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

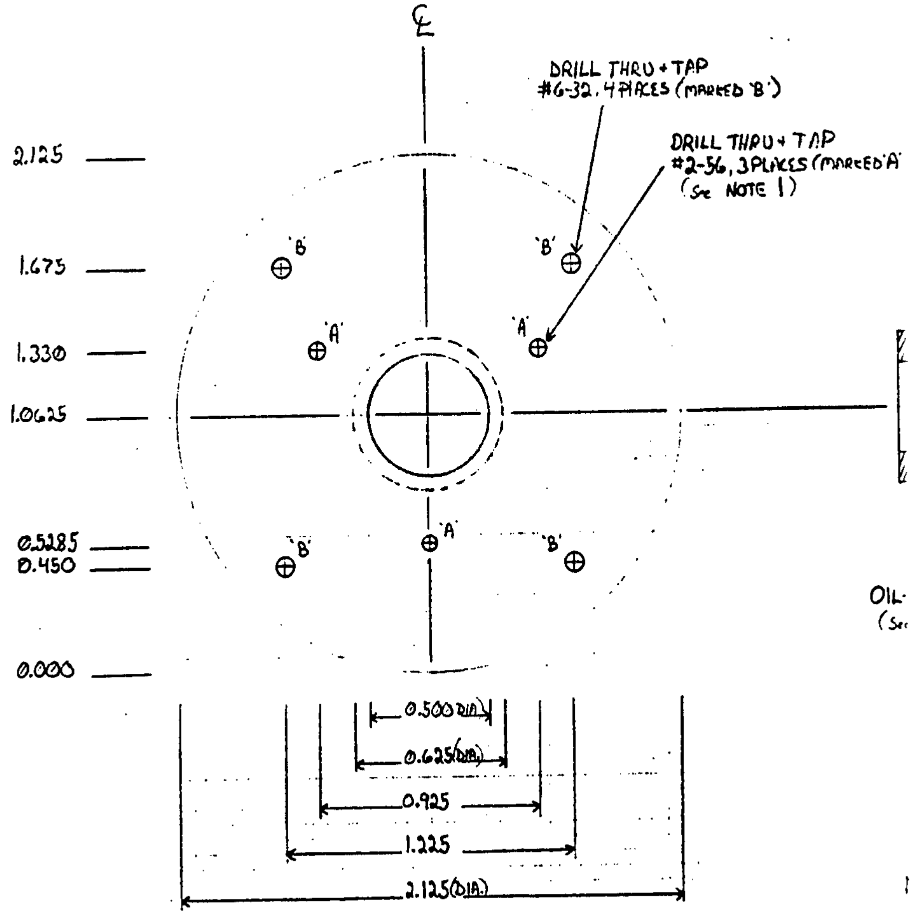
MBA EXOSKELETON
VANE ACTUATOR ADAPTOR SHAFT

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE X2 RELEASE DATE SHEET | OF

2 1

2



QTY REQD PER ASSY

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

PART OR IDENTIFYING NO. | C

UNLESS OTHERWISE SPECIFIE
DIMENSIONS ARE IN INCHES

TOLERANCES:

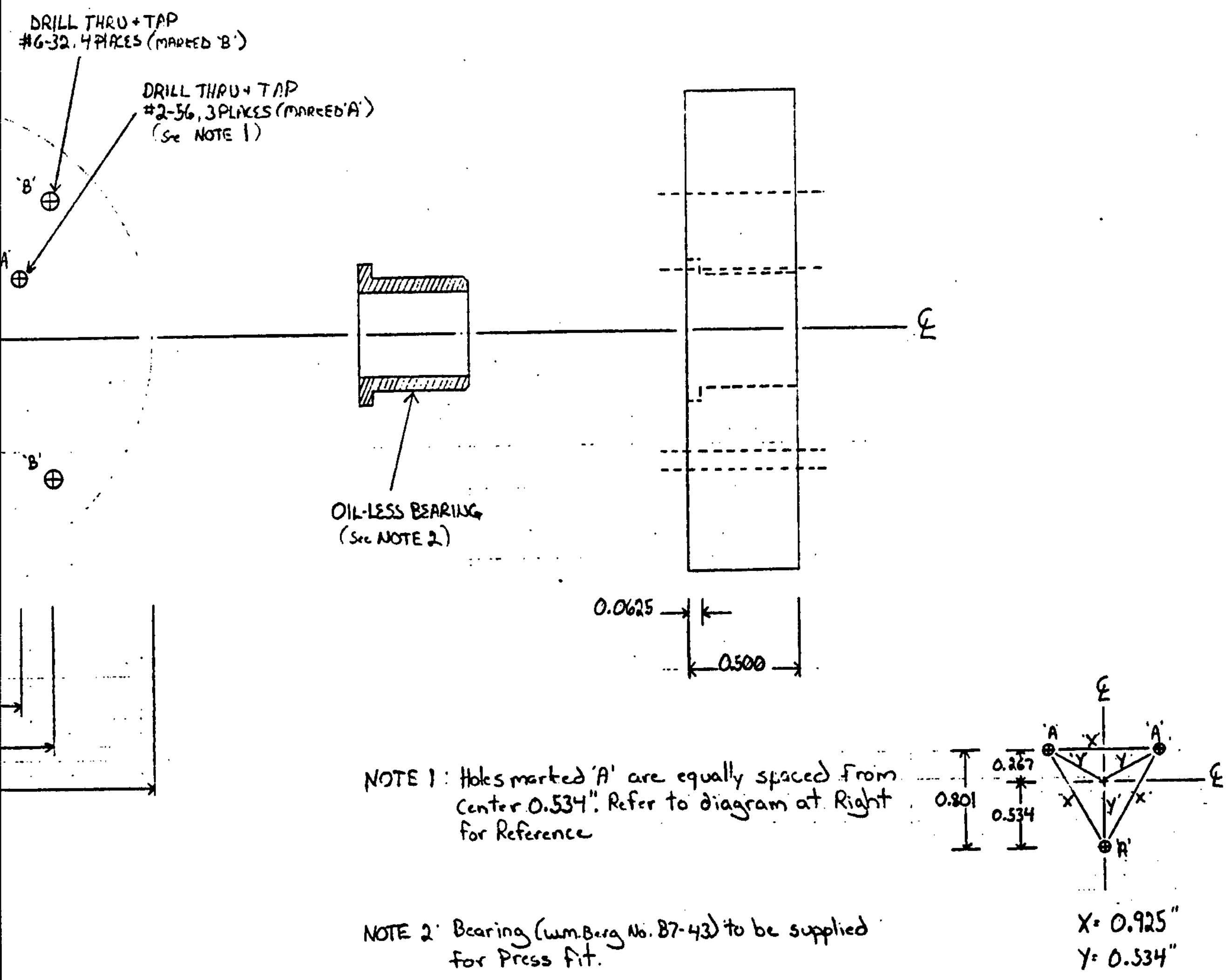
.XX = ANGLES =
.XXX = FRACTIONS = ±
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL
6061-T6 Aluminum

FINISH
Black Anodize

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED



NOTE 1: Holes marked 'A' are equally spaced from Center 0.534". Refer to diagram at Right for Reference.

NOTE 2: Bearing (wm. Berg No. B7-43) to be supplied for Press fit.

		PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:
XX =
XXX =
XXXX = BASIC
ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL 6061-T6 Aluminum

FINISH Black Anodize

USED ON

CONTRACT NO.		
DRAWN BY	dLogan	DATE 12-28-87
CHECKED		
DESIGN		
PROJECT	5426-60-27	
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELETON
VANE ACTUATOR ADAPTOR PLATE

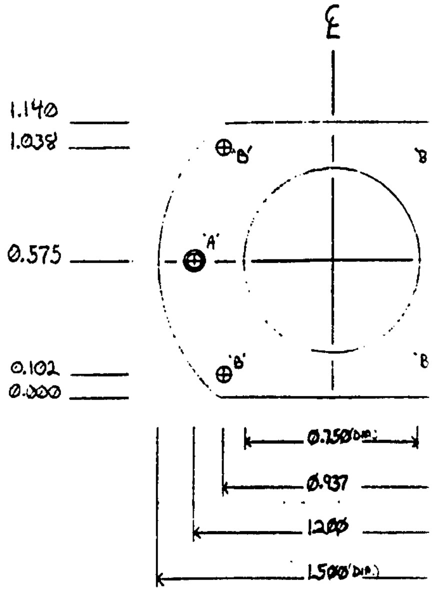
SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE X2 RELEASE DATE SHEET 1 OF 1

4 3

D

C



B

A

QTY REQD PER ASSY				
PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

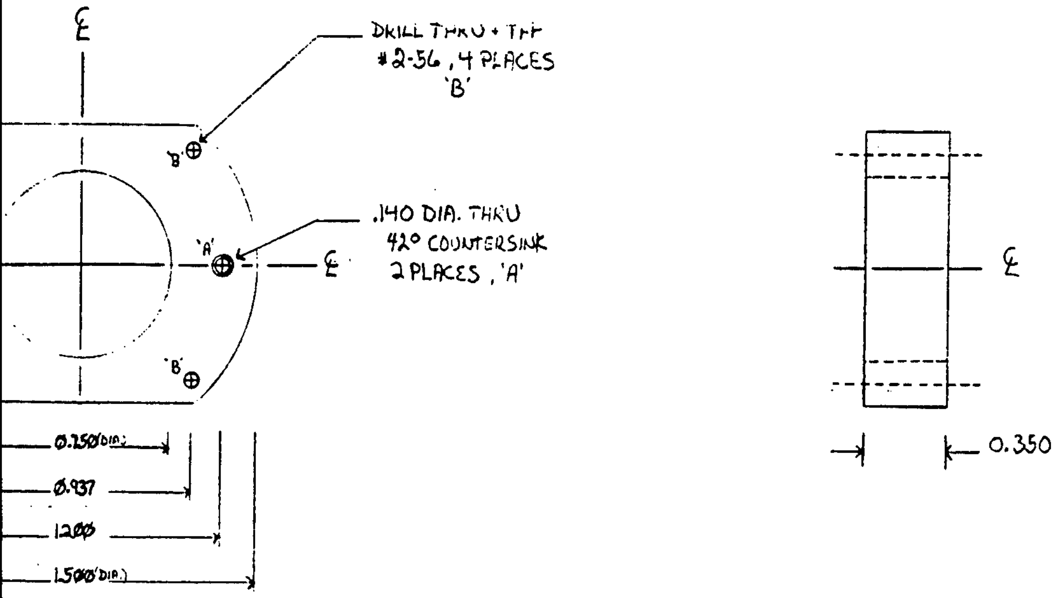
MRC

4 3

3 2 1

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

D



C

B

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX =
.XXX =
.XXXX = BASIC
ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL
6061-T6 Aluminum

FINISH
Black Anodize

USED ON

CATION

CONTRACT NO.		
DRAWN BY	J. Logan	DATE 12-28-87
CHECKED		
DESIGN		
PROJECT	5426-60-27	
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

ENCODER MOUNTING BRACKET,
UPPER + LOWER ROLL JOINTS (U+L)

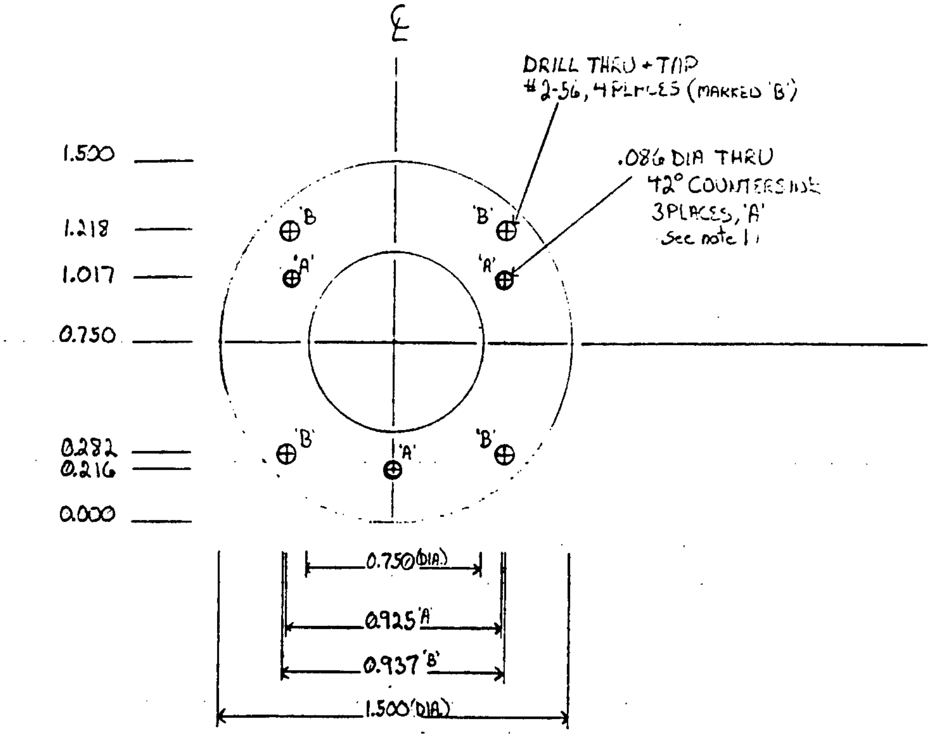
A

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE x2 | RELEASE DATE | SHEET 1 OF 1

2 1

2



NOTE

QTY REQD PER ASSY

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

PART OR IDENTIFYING NO. | CODE IDENT | NOMENCL

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX =
.XXX =
.XXXX = BASIC

ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL 6061-T6
Aluminum

FINISH Black Anodize

CONTRACT NO.

DRAWN BY

CHECKED

DESIGN

PROJECT

CUSTOMER

QUALITY ASSURANCE

MANUFACTURING

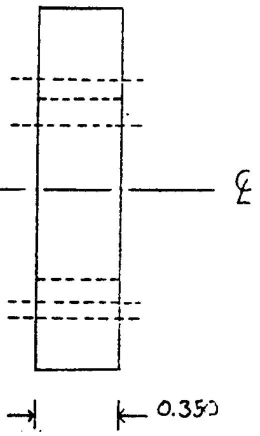
2 | 1

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

D

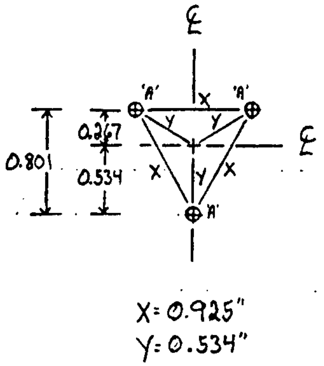
P
(MAKKED 'B')

IA THRU
COUNTERSINK
ACES, 'A'
e note 1



C

NOTE 1 Holes marked 'A' are equally spaced from center 0.534 in. Refer to diagram at right.



B

.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

ECIFIED
NCHES

LES = ± 30'
NS = ± 1/32

✓

ze

CONTRACT NO.		
DRAWN BY	J Logan	DATE 12-28-87
CHECKED		
DESIGN		
PROJECT	5426-60-27	
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

ENCODER MOUNTING BRACKET,
STANDARD JOINTS (A,S,R,F,E,G)

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

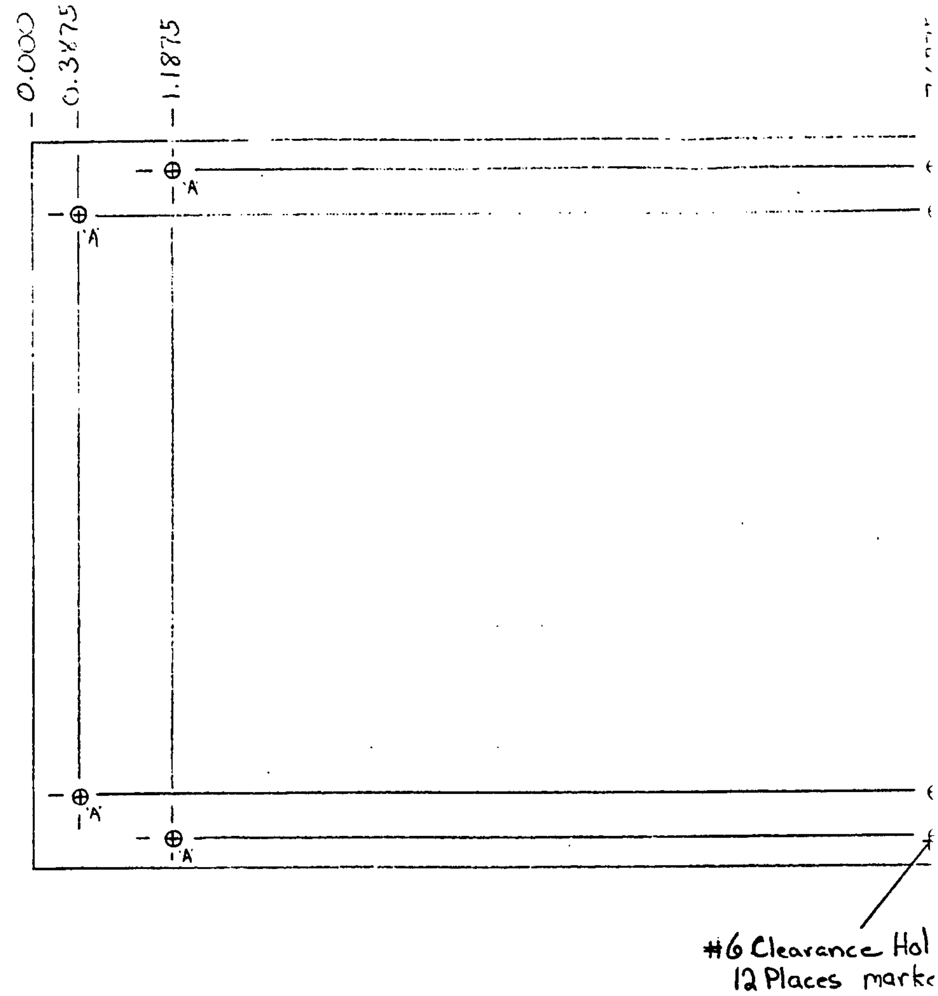
SCALE: X 2 | RELEASE DATE | SHEET 1 OF 1

A

2

4 3

D



C

B

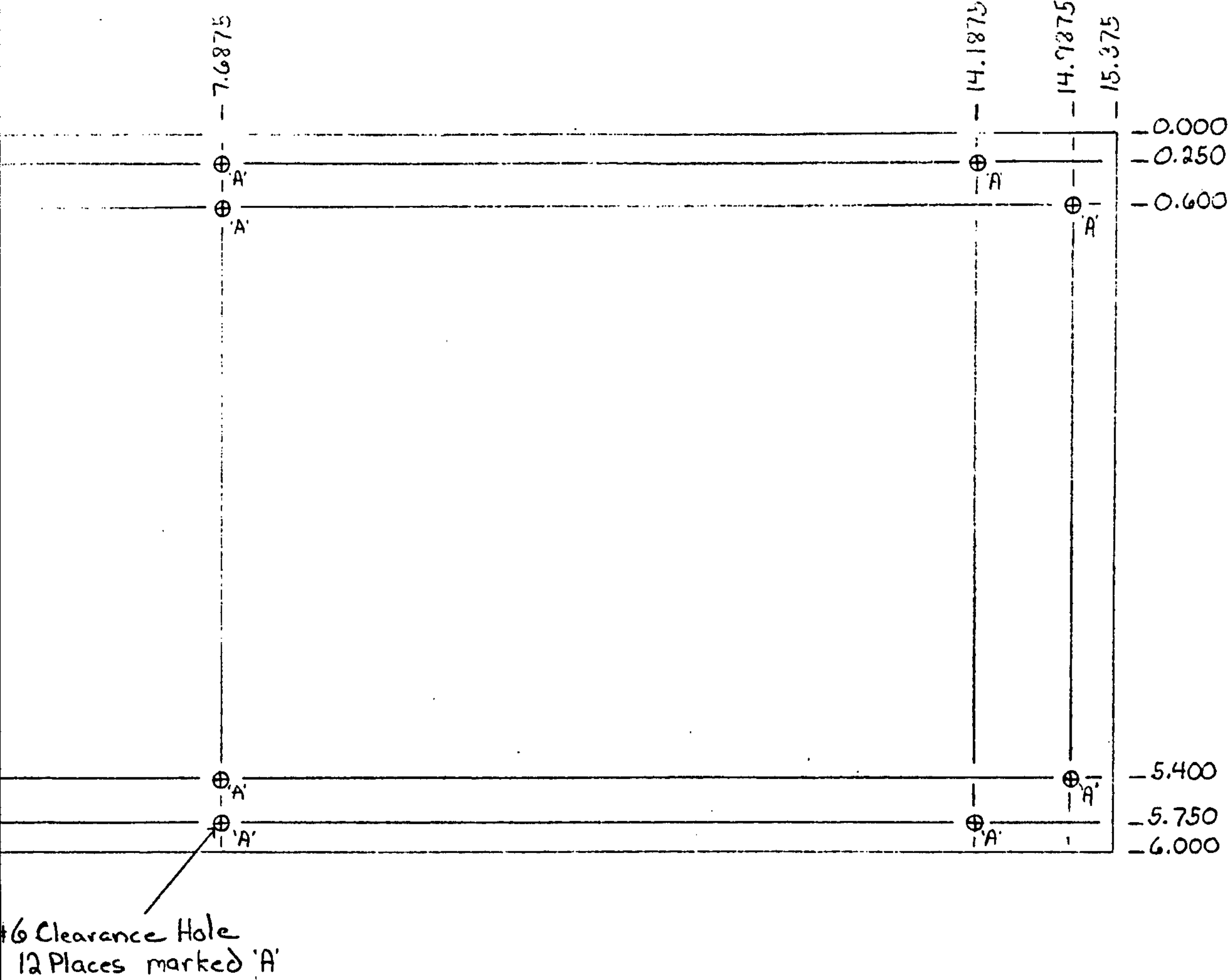
A

QTY REQD PER ASSY					PART
					UN. D.
					XX = XXX = XXXX
					MATER
PART NO.	N/A	F/A	NEXT ASSY	USED ON	
	QTY REQD		APPLICATION		FINISH

MRC

4 3

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED



#6 Clearance Hole
12 Places marked 'A'

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:
.XX =
.XXX =
.XXXX = BASIC
ANGLES = ± 30'
FRACTIONS = ± 1/32

ALL SURFACES ✓

MATERIAL .062 Aluminum

FINISH Black Anodize

USED ON

CATION

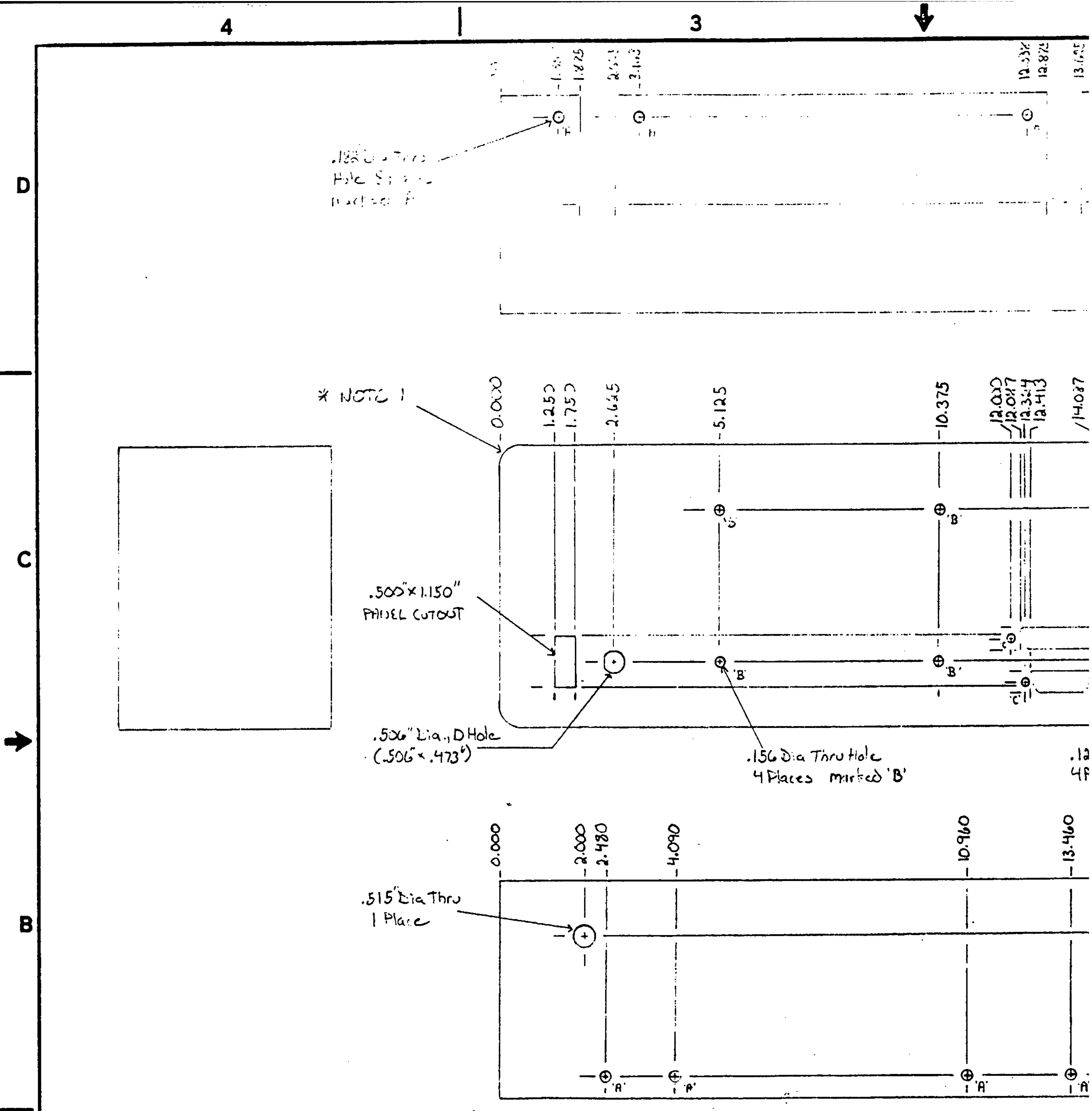
CONTRACT NO.		
		DATE
DRAWN BY	J.Logan	5-24-88
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELETON
BACKPACK BASEPLATE

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE | RELEASE DATE | SHEET OF



QTY REQD PER ASSY

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

PART OR IDENTIFYING NO.	CODE IDENT	N

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL
.062 Aluminum

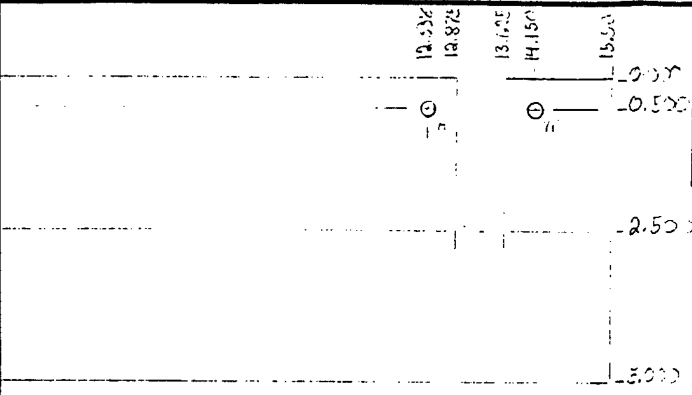
FINISH
BlACK Anodize

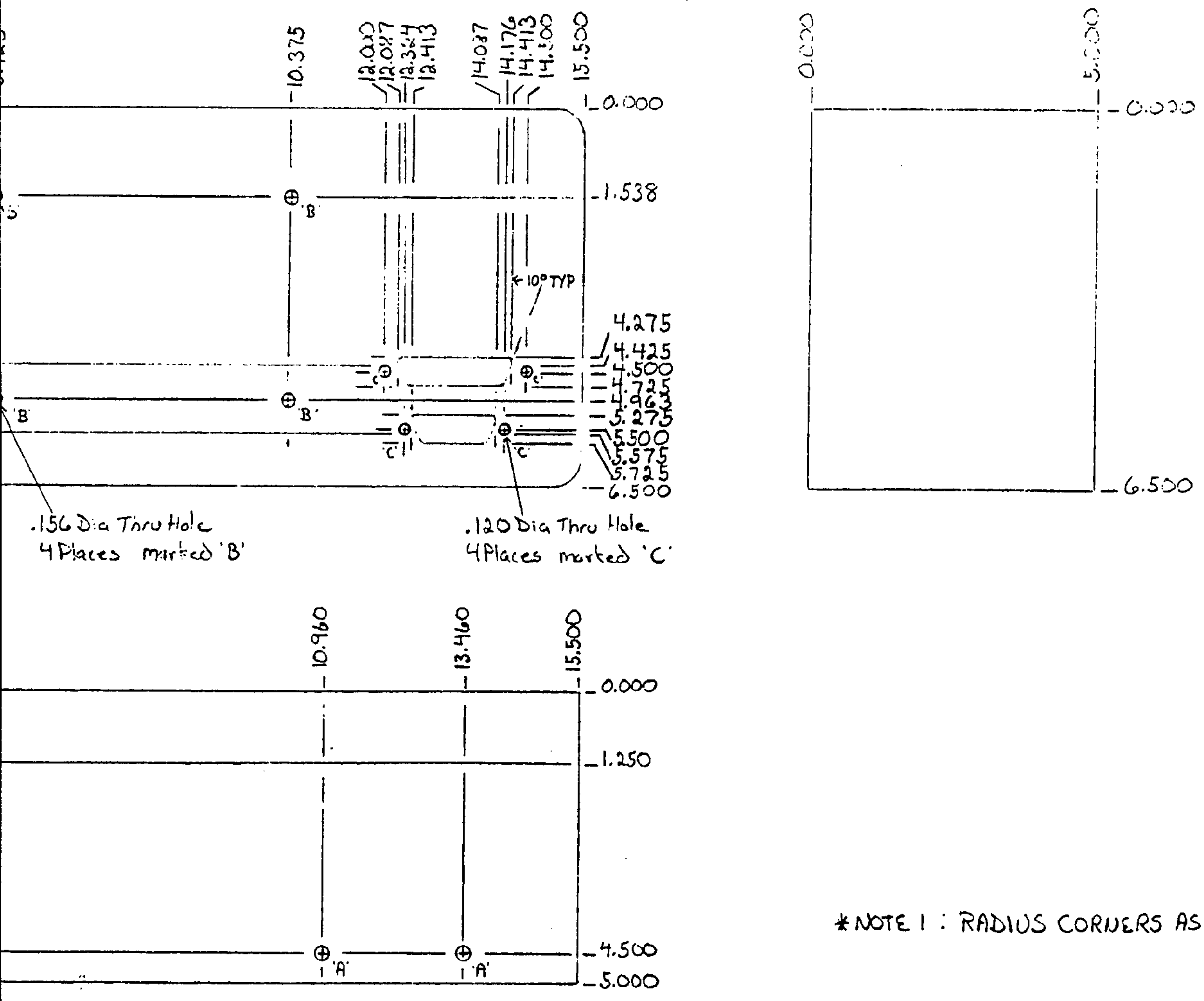
CONTRA
DRAWN I
CHECKEI
DESIGN
PROJECT
CUSTOM
QUALITY ASSURAI
MANUFA

4 3

MRC

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED





*NOTE 1 : RADIUS CORNERS AS MUCH AS POSSIBLE

PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.
PARTS LIST						

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

XX = ANGLES = ± 30'
XXX = FRACTIONS = ± 1/32
XXXX = BASIC

ALL SURFACES ✓

MATERIAL
.062 Aluminum

FINISH
BlACK Anodize

USED ON

CONTRACT NO.		
DRAWN BY	J. Logan	DATE 5-24-88
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

MBA EXOSKELETON COVER,
BACK-PACK ELECTRONICS

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE RELEASE DATE SHEET OF

2

9.0 Appendix A: MBAssociates Backpack

9.4 Software Source Listing

mba.c68

```
/*****************************************************

  FILE      mba.c68

  FUNCTION   main

  AUTHOR    Todd Mosher

  DATE      5-6-91

  GENERAL DESCRIPTION
  This program runs in the mba back pack.
  This routine will wait for the user to
  move the joints through their indexs and
  then it will return the position of the encoders
  with respect to their index position.

*****************************************************/
#include <stdio.h68>
#include <ctype.h68>
#define ESTOP 0x0024
#define GOHALT 0x0009

#define chr_in() ((iostat[0]) & 0x01)     /* check char avail status */
#define get_byt() (ioport[0])            /* look at the incoming char */
        int *optenc;                  /* pointer to optical encoder values */
        int *optsta;                  /* pointer to optical encoder status */
        char *ioport;
        char *iostat;
        int  *swt;
        int *lights;

        int glob_pos[16];              /* position from index mark */
        int last_pos[16];             /* last encoder val read */
        int reqpnd;
        int send();
        int init_oe();
        int position();

main()
{
        int comcnt = 0;
        int i,j;
        int tog = 0;
        int off = 0xffff;
        int on = 0xfefe;          /* only gripper light on */
        int lit_mask;

     for (i=0;i<16;i++)
     {
      glob_pos[i] = 0;
      last_pos[i] = 0;
    }
    optsta = 0x24020;
    ioport = 0x240e7;
    iostat = 0x240e3;
    swt = 0x24080;
    lights = 0x24040;     /* point to lights */
    optenc = 0x24000;     /* point to optical encoders */

reqpnd = 0;
init_oe();           /* initialize optical encoders */
lit_mask = on;
*lights = lit_mask;
while (1)            /* do forever */
{
      if (chr_in() || reqpnd) /* if char in the serial io  port */
      {
            if (comcnt++ == 100)       /* show comm with blinking lights */ lit_mask = lit_mask &
                  0xefef;
               if (comcnt == 200)
               {
                  lit_mask = on;
                  comcnt = 0;
               }
               tog = ~tog;
               if (tog)
                  lit_mask = lit_mask & 0x7f7f;
               else
                  lit_mask = lit_mask | 0x8080;
               *lights = lit_mask;

            if (!reqpnd)            /* if a request was pending */
                        i = get_byt();
                 else
                        i = reqpnd;

            if (i == 'H' || i == 'B' || i == 'L' ||
                 i == 'R' || reqpnd)          /* pc requesting data */
               {
                  reqpnd = 0;
                  if ((swt[0] & GOHALT) != GOHALT)   /* z for sleep/halt */ send("Z",1);
                  else if ((swt[0] & 0x0012) == 0x0012)   /* no index swt pressed */ send("H",1);
                  else if ((swt[0] & 0x0012) == 0)   /* both indexs  pressed */
                        send("R",1);                /* reset indexes */
                    else
                        send("I",1);                /* indexing mode */

                  if (i == 'H' || reqpnd == 'H')
```

```
                            for (i=0;i<16;i++)     /* send out all data */
                    {
                        send(&glob_pos[i],2);  /* return an error code */
                        position();         /* get new encoder positions */
                      }
                  else if (i == 'L' || reqpnd == 'L')
                    for (i=0;i<8;i++)     /* send out all data */
                    {
                        send(&glob_pos[i],2);  /* return an error code */
                        position();         /* get new encoder positions */
                      }
                  else if (i == 'R' || reqpnd == 'R')
                    for (i=8;i<16;i++)     /* send out all data */
                    {
                        send(&glob_pos[i],2);  /* return an error code */
                        position();         /* get new encoder positions */
                      }
                }
                else if (i == 'S')           /* pc requesting status */
                send("S",1);

                         position();              /* get new encoder positions */
        }
/*    if ((swt[0] & ESTOP) != ESTOP)    /* allow user to get out on */
/*     exit(0);                    /* emergency stop */

/*    while ((swt[0] & GOHALT) != GOHALT)  /* send nothing for gohalt */
        position();                /* keep track of position */
      }
      i = 1;
      if (!i)
        printf(" ");       /* this is because of a bug in the libs */
}

int position()
{
      int cur_pos;
      int i;

    for (i=0;i<16;i++)    /* for all encoders, calc their positions */
    {
        cur_pos = optenc[i];
        if (cur_pos >= last_pos[i])   /* clock wise or else ccw past index */
            if ((last_pos[i] + 2046) > cur_pos)  /* cw motion */
               glob_pos[i] += cur_pos - last_pos[i];
             else
               glob_pos[i] -= last_pos[i] + 4096 - cur_pos;/* ccw past index*/
            else               /* ccw or else clock wise past index */
            if ((cur_pos + 2046) >= last_pos[i])   /* ccw motion */
               glob_pos[i] -= last_pos[i] - cur_pos;
             else
        glob_pos[i] += cur_pos + 4096 - last_pos[i]; /* cw through index */ last_pos[i] = cur_pos;
       }
int init_oe()             /* initialize the optical encoders */
{                        /* set bits to 1 which will cause the */
      int list[16];         /* hw to reset counters when the index is */
      int tmp;              /* crossed */
      int i;
      int j;

   for (i=0;i<16;i++)
        list[i] = 0;

   *lights = 0;              /* turn all lights on */
   *optsta = -1;              /* set all bits to 1 to allow the hardware */
   tmp = optsta[0];            /* to reset the counters when the index is passed */
/*  while(optsta[0])          /* while not all indexs crossed */
   while(optsta[0] & 0xff00)  /* while not all indexs crossed */
   {
        *lights = ~(*optsta); /* lights reflect the status to users */
        if ((swt[0] & 0x0012) != 0x0012)  /* index switch aborts */
        {               /* this is incase of broken encoder */
            optsta[0] = 0;
            for (i=0;i<16000;i++);  /* delay for swt release */
            break;
          }
          if (tmp != *optsta)   /* 1 or more indexs crossed */
          {
            j = 1;
            for (i=0;i<16;i++)  /* which ones were crossed */
            {
                if ((*optsta & j == 0) && !list[i])  /* new index crossed */
                {
                    list[i]++;                  /* note that index crossed */
                    glob_pos[i] = optenc[i];      /* set global position info */ last_pos[i] = glob_pos[i];
                  }
                  j = j<<1;
              }
              position();
          }
          if (chr_in())     /* if there is a char in the serial io  port */
          {
            i = get_byt();
            if (i == 'H' || i == 'B' ||
                        i == 'L' || i == 'R')          /* pc requesting data */
                       reqpnd = i;          /* note the pending request  */
                 else if (i == 'S')      /* pc requesting status */
                send("S",1);
            }
```

```
    }
}
int send(byt,cnt)          /* send cnt bytes out the serial io port */
char *byt;                /* if cnt is 0 then an ascii string is being */
int cnt;                 /* sent */
{
       int i;
       long j;

    if (!cnt)             /* sending a text string */
        cnt = strlen(byt);
      for (i=0;i<cnt;i++)     /* send out all bytes */
      {
        for (j=0;j<64000l;j++)    /* wait for tx free */
            if ((iostat[0] & 0x04) == 0x04)  /* tx ready */
                break;
            if (j < 64000l)          /* max wait for tx ready */
            ioport[0] = byt[i];
          else
            return(1);    /* return error */
        }
        return(0);
}
```

makcur.bat

```
c68  +fi -o mba.r68 mba.c68
ln68 +c 1000 -t -o mba.e68 mba.r68  -lc68s
```

setenv.bat

```
set INCL68=c:\aztec_c\include
set CLIB68=c:\aztec_c
```

exomon.c

```
"C"
"68000"
#$EXTENSIONS ON$
#$FULL_LIST ON$
#$ENTRY OFF$
#$LIST_CODE ON$

/*****************************************************************/
/*                                                               */
/*  DESIGNED BY : TODD MOSHER / MONTY CRABILL                    */
/*                                                               */
/*  CODED BY : TODD MOSHER                                       */
/*                                                               */
/*  DATE : JAN-16-1991                                           */
/*                                                               */
/*  REVISION NUMBER: ORIGINAL                                    */
/*                                                               */
/*****************************************************************/
/*                                                               */
/*  REVISIONS LOG:                                               */
/*  NUMBER   DATE          DESCRIPTION                           */
/*  ------   ----          -----------                           */
/*                         RL_C_MBA                              */
/*****************************************************************/
/*                                                               */
/*  THIS PROGRAM IS FOR THE MBA BACKPACK.  IT IS A MONITOR       */
/*  WHICH WILL INITIALIZE THE ENCODERS AND THEN WAIT FOR A       */
/*  FILE TO BE UPLOADED OVER THE SERIAL PORT.  DURING            */
/*  INITIALIZATION OF THE ENCODERS, ALL LEDS WILL BE LIT.        */
/*  AT THIS POINT, ALL JOINTS MUST BE ROTATED THROUGH THEIR      */
/*  RANGE OF MOTION SO THAT THE LEDS WILL GO OFF.                */
/*  THEN THE LEDS WILL CONTINUE TO FLASH UNTIL A FILE STARTS     */
/*  UPLOADING.  WHILE THE FILE IS LOADING, THE LIGHTS WILL DO    */
/*  A COUNTING PATTERN, AND FINALLY THEY WILL BE TURNED OFF      */
/*  WHEN THE UPLOADED PROGRAM BEGINS EXECUTION.                  */
/*  ERROR CODES WILL BE DISPLAYED ON THE LEFT HAND               */
/*  ERROR CODES:                                                 */
/*    LIGHT      ERROR                                           */
/*     GP      "PGM" PREFACE TO FILE TX MISSING                  */
/*     WF       STARTING ADDR OF CODE MISSING                    */
/*     WR       CODE SIZE MISSING                                */
/*     LR      STARTING ADDR OF DATA MISSING                     */
/*     EB      DATA SIZE MISSING                                 */
/*     UA       NOT ENOUGH CODE SENT UP                          */
/*     SE      NOT ENOUGH DATA SENT UP                           */
/*     SA      CHECK SUM ERROR                                   */
/*                                                               */
/*   SEE UPLOAD FUNCTION HEADER FOR EXPECTED INFO FOR FILE TX    */
/*                                                               */
/*****************************************************************/
```

exomon.c

```
#define DUART_A 0x0240E7
#define DUART_B 0x0240F7
#define TXR_MASK 0x04
#define SND_MASK 0x01
/*char *jstack = 0x20000; /* FOR EPROM 0-3 MUST BE STACK POINTER */ int main();
/*char *jaddr = main;  /* FOR EPROM... 4-7 MUST BE PGM PC ADDR */
                        /* MUST BE 20100  IN EPROM  BITS 4-7 */
/*char *jaddr = 0x30228;     FOR EMULATION */
extern int STRT();
#$ORG 0x24080$
int switches;        /* gripper switches   bit 0 - left halt/go */
             /*                   bit 1 - left indx    */
             /*                   bit 2 - left E stop  */
             /*                   bit 3 - right halt/go*/
             /*                   bit 4 - right indx   */
             /*                   bit 5 - right E stop */
         /* note debug is bit 0 @ 0x24060 */
#$END_ORG$
#$ORG 0x240E3$
short txr_stat_ptr;
#$END_ORG$
#$ORG 24020H$
  int oe_sta;

#$END_ORG$
short *dev_ptr;
char tarr[4];
char (*usrpgm)();          /* pointer to user program  */
int    glberr;          /* general purpose error indicator */
int *grp_led;            /* addr of griper leds */
int grpdsp;
char *cadr;
char sndstr[100];       /* string to send from function send */
int wait();            /* waits for tx ready */
int send();
int getbyt();           /* byte from serial port, -1 = time out */
/* int init_oe();          /* optical encoder initialization */
int upload();        /* attempts to upload and execute the usr pgm */

main()
{
        short z;
        short j;
        int   k;
        short *oe_ptr,i;
        short *c_r_ts;
      grp_led = (int *)0x24040;
      grpdsp = 0;
      c_r_ts = (short *)0x240E1;       /* setup pointers */
      dev_ptr =(short *)DUART_A;
```

exomon.c

```
    STRT();             /* CALL STRT TO INITIALIZE SERIAL PORT */
    glberr = 0;

    *c_r_ts = 0x13;
    *c_r_ts = 0x1F;

    k = 0;
    while(1)        /* wait for a program to be uploaded */
    {
        if ((switches & 0x0012) != 0x0012) /* index clears the err flag */ glberr = 0;

        if (!glberr)
            k++;
          if (k == 20000 && !glberr)    /* if no error, flash leds */
          {
            if (grpdsp)
                grpdsp = 0;
              else
                grpdsp = 0xffff;
              *grp_led = grpdsp;
              k = 0;
          }

        j = (short)(txr_stat_ptr & SND_MASK);  /* get rs422 status */
        if (j == 1)                              7* char rx from PC */
        {
            *grp_led = 0xffff;     /* turn lights off */
            if (upload() != 1)     /* upload and exec user program */
                send("ER");         /* on error, send error msg */
            }
      }
}
```

exomon.c

```
/****************************************************************/
/*                                                              */
/*  THIS ROUTINE WILL CONTROL THE UPLOADING OF A PROGRAM        */
/*  THIS ROUTINE EXPECTS                                        */
/*     BYTES 0-2   PGM                                          */
/*     BYTES 3-6   CODE START ADDR                              */
/*     BYTES 7-10  BYTES OF CODE                                */
/*     BYTES 11-14 DATA START ADDR                              */
/*     BYTES 15-18 BYTES OF DATA                                */
/*     CODE DATA                                                */
/*     DATA DATA                                                */
/*     1 BYTE CHECK SUM                                         */
/*                                                              */
/*  THE PC UPLOAD PROGRAM ALWAYS SENDS OUT AN 'S' BEFORE        */
/*  STARTING THE UPLOAD.  THIS IS TO STOP ANY PROGRAMS WHICH    */
/*  MAY ALREADY BE RUNNING IN THE BACK PACK                     */
/*                                                              */
/*  AFTER EACH THING LISTED ABOVE, AN 'OK' OR 'ER' WILL         */
/*  BE SENT TO THE SENDING DEVICE, ON 'ER', THIS ROUTINE        */
/*  WILL RETURN TO MONITOR MODE                                 */
/*                                                              */
/****************************************************************/
int upload()
{
    long i,j;
    char chksum = 0;
/*  char *cadr = 0l;          /* code address */
    long csiz  = 0;           /* size of code in bytes */
    char *dadr = 0l;          /* data address */
    long dsiz  = 0;           /* data size */
    long tlong;
    long tmp;
    int  err = 0;

    i = getbyt();
    if (i == 'S')
        return(1);            /* ignore initial S's */

  if (i      != 'P' ||        /* must start with PGM */
     getbyt() != 'G' ||
     getbyt() != 'M' )
    return(glberr = (*grp_led = 0xfffe));  /* return error */
  send("OK");
  cadr = 0l;
  tlong = 0l;
  for (i=3;i>=0;i--)              /* get code start address */
    if ((tarr[i] = getbyt()) == -1)     /* -1 = no data available
      */ return(glberr = (*grp_led = 0xfffd));
    else
      tlong = tlong * 256 + tarr[i];  /* build code address */
  cadr = (char *)tlong;
```

exomon.c

```
send("OK");

for (i=3;i> =0;i--)              /* get code size in bytes */
    if ((tmp = getbyt()) = = -1)     /* -1 = no data available */ return(glberr = (*grp_led =
        0xfffb));
      else
        csiz   = csiz  | (tmp < < (8 * i));/* build code size */

send("OK");

for (i=3;i> =0;i--)              /* get data start address */
    if ((tmp = getbyt()) = = -1)     /* -1 = no data available */ return(glberr = (*grp_led =
        0xfff7));
      else
        tlong = tlong | (tmp < < (8 * i));  /* build data address */ dadr = (char *)tlong;

send("OK");

for (i=3;i> =0;i--)              /* get data size in bytes */
    if ((tmp = getbyt()) = = -1)     /* -1 = no data available */ return(glberr = (*grp_led =
        0xffef));
      else
        dsiz  = dsiz  | (tmp < < (8 * i));  /* build data size    */

send("OK");

for (i=0;i<csiz;i+ +)           /* load in all code */
    if ((tmp = getbyt()) = = -1)
        return(glberr = (*grp_led = 0xffdf));
      else                     /* no error so continue */
      {
        if (grpdsp+ + >32760)
            grpdsp = 0;
          *grp_led = grpdsp;
          chksum += tmp;
          cadr[i] = tmp;
      }
  send("OK");
  for (i=0;i<dsiz;i+ +)           /* load in all data */
    if ((tmp = getbyt()) = = -1)
        return(glberr = (*grp_led = 0xffbf));
      else                     /* no error so continue */
      {
        if (grpdsp+ + > 32760)
            grpdsp = 0;
          *grp_led = grpdsp;
          chksum += tmp;
          dadr[i] = tmp;
      }
  send("OK");
```

```
if (chksum == getbyt())    /* verify check sums of code&data */
     {
         send("OK");
         *grp_led = 0xffff;        /* turn leds off */
/*   while ((switches & 0x0012) == 0x0012);  DEBUG ONLY */
         usrpgm = cadr;            /* set up jump to user program */
         usrpgm();               /* go execute  user program */
       }
       else
         return(glberr = (*grp_led = 0xff7f));
       return(1);                 7* all done, go back to monitor */
}


/****************************************************************/
/*                                                              */
/*  THIS ROUTINE WILL SEND OUT THE REQUESTED DATA FROM THE     */
/*  BUFFER.  (LATER ON THIS SHOULD BE CONVERTED INTO AN        */
/*  INTERRUPT DRIVEN ROUTINE)                           */
/*                                                              */
/****************************************************************/
int send(str)
char *str;
{
    int i;

     for (i=0;str[i] != 0;i++)       /* for all data points */
     {
         wait();                    /* wait for tx ok status */
         *dev_ptr = str[i];           /* send the data byte  */
       }
}

/****************************************************************/
/*                                                              */
/*  THIS ROUTINE WILL WAIT FOR AN INCOMMING CHARACTOR FROM     */
/*  THE SERIAL PORT.   THIS SENDS BACK THE CHAR BEING RX AND   */
/*  WILL SEND BACK A -1 ON TIME OUT.  THE TIMEOUT CHOSEN IS    */
/*  SOME WHAT A RANDOM CHOICE WHICH HAS BEEN PROPERLY ADJUSTED */
/*  DURING TESTING.                                  */
/*                                                              */
/****************************************************************/
int getbyt()
{
    int i,j,ch_in;
    int status;
  for (i=0;i<12000;i++)
  {
     for (j=0;j<50;j++)
       if ((txr_stat_ptr & SND_MASK) == 1)  /* char rx */
         break;
```

```
                    if (j<50)                         /* yes char rx */
            break;
        }

    if (i < 12000)                     /* yes char rx */
        return(*dev_ptr & 0x00ff);           /* return char */
      return(-1);                      /* else no char rx */
}


/*****************************************************************/
/*                                                              */
/*  THE ROUTINE WAIT POLLS THE XMIT BUFFER STATUS WAITING FOR  */
/*  A FREE TRANSMIT BUFFER.                                     */
/*                                                              */
/*****************************************************************/
int wait()
{
  while(txr_stat_ptr & TXR_MASK != 4);
}

/*****************************************************************/
/*                                                              */
/*  THE ROUTINE INIT_OE MAPS THE OPTICAL ENCODERS IN THE        */
/*  MEMORY.  THE ROUTINE THEN WAITS FOR EACH OF THE OPTICAL     */
/*  ENCODERS TO BE MOVED THROUGH THEIR ZERO POINTS.  UPON       */
/*  COMPLETION OF THIS INITIALIZATION PROCEDURE CONTROLL IS     */
/*  RETURNED TO THE CALLING ROUTINE.                            */
/*    ----- not used any more - this taken care of by uploaded */
/*         program   -------                                    */
/*****************************************************************/
int init_oe()
{
#$ORG 24000H$
  int           lgp_cnt;
  int           lwf_cnt;
  int           lwr_cnt;
  int           llr_cnt;
  int           leb_cnt;
  int           lua_cnt;
  int           lse_cnt;
  int           lsa_cnt;
  int           rgp_cnt;
  int           rwf_cnt;
  int           rwr_cnt;
  int           rlr_cnt;
  int           reb_cnt;
  int           rua_cnt;
  int           rse_cnt;
  int           rsa_cnt;
```

```
    *grp_led = 0;
    oe_sta = -1;
    do
    {
        *grp_led = ~oe_sta;
        if ((switches & 0x0012) != 0x0012) /* user may abort - indx sw */ break;
      }
      while (oe_sta);
      *grp_led = -1;
}
```

10.0 Appendix B: MBA/Merlin Control Software Listing

merlin.prj

```
merlin
testsys
getjr3
prockey
utime
keycont
movearm
calibmer
mercmds
window
forkin
sinvkin
merinit
merltmat
utils
enctorad
fullsys
transp
wristang
merlrmat
jointang
mbainit
rexotmat
calib
getmba
ttyopen
ttylisr.obj
tmwinl.lib
lwin.lib
```

merlin.c

```
/*********************************************

    FILENAME:  Merlin.C

            PROGRAM:   (main)

            AUTHOR:    TW Mosher  ML Crabill

    DATE:      4 April 91

    DESCRIPTION:

*****************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.def"

extern int forkin();     /* forward kinematic function */
int cbreak()             /* control the ctrlc vector */
{
    wn_exit();
    tty_close();
    exit(printf("User aborting program\n"));
}
main()
{
        long tst[6] = {0};
        int j,i,k,l;

    window_init();     /* bit 3 window init */
    wn_init();         /* graphics screen init */
    mer_init();        /* variable initialization */
    clrscr();
    ctrlbrk(cbreak);   /* setup ctrl c vector */
    mer_init_serv();   /* init servo values - gain ... */

    while(1)
    {
                                /* fill in main menu */
                fill_menu(25,5," HSHI ",
                        "CALIBRATE          ",
                        "DIAG FORWD KIN     ",
                        "INVERSE KIN        ",
                        "MOVE TO ZERO POS ",
                        "FULL SYSTEM        ",
                        "TEST SYSTEM        ",
                        "ZZZ - do not use ",
                        "EXIT & ZERO POS   ",
                        "TERM HSHI & EXIT ",
                        "QUIT               ",
                    NULL);  /* must end with null */

switch(menu())      /* display menu and get option selected */
{
case 0:             /* calibrate */
    calib_mer();
    break;
  case 1:           /* do fdw din */
    mer_init_serv();  /* set 0 deg here */
    j = status(1,11," FORWARD KINEMATICS INFO ",75,12);
    key_control(forkin,&j);
    close_status();
    break;
  case 2:           /* do merlin inverse kinematics */
    mer_init_serv();  /* set 0 deg here */
    j = status(1,8," INVERSE KINEMATICS INFO ",75,12);
    while(1)
    {
        fill_form(1,1,"X POSITION   "," ",-1,-1,'D',6,&x_pos);
        fill_form(1,3,"Y POSITION   "," ",-1,-1,'D',6,&y_pos);
        fill_form(1,5,"Z POSITION   "," ",-1,-1,'D',6,&z_pos);
        form_disp(" TOOL TIP POSITION "," ",1,1,
                                         25,3,25,8);  /* display initial form */
                              form_exe(&i,0);  /* get user input for xyz */
                              form_close();
                              /* invkin(j,x_pos,y_pos,z_pos);  */
                              sinvkin(j,x_pos,y_pos,z_pos);
                              i = wind(20,5,"DO MORE?",'V','Y',"Enter more points (y)",NULL)
                              if (i != 'Y')
            break;
        }
        close_status();
        break;
  case 3:           /* move joints to zero position */
    mer_close();            /* sets joints to 0 position */
    break;
  case 4:           /* full exo master to merlin slave system */
    mer_init_serv();  /* set 0 deg here */
    full_system();    /* full up control */
    mer_close();
    break;
  case 5:           /* full test system */
    mer_init_serv();  /* set 0 deg here */
    test_system();    /* full up control */
    mer_close();
    break;
  case 6:           /* test code */
    mer_init_serv();  /* set 0 deg here */
    i = 0;
    j = 0;
    k = 0;
```

```
             while (1)
             {
                 fill_form(1,1,"option ","  ",-1,-1,'I',1,&k);
                 fill_form(1,3,"deg   ","  ",-1,-1,'I',3,&j);
                 form_disp(" max joint tst ","  ",1,1,
                                       25,3,25,8);  /* display initial form */
                            form_exe(&i,0);   /* get user input for xyz */
                            form_close();
                            if (k > 1)
                     break;
                   if (k == 0)   /* option move to deg */
                   {
                     for (i=0;i<3;i++)
                         tst[i] = j * 276;
                       movearm(tst,'E');
                   }
                   else        /* move a deg each time */
                   {
                     for (i=0;i<3;i++)
                         tst[i] = 0;
                       for (i=0;i<j;i++)
                         for (l=0;l<3;l++)
                         {
                             tst[l] += 276;
                             movearm(tst,'E');
                           }
                       }
                       wind(2,2,"pause",'E',' ',"  ",NULL);
                       for (i=0;i<3;i++)
                         tst[i] = 0;
                       movearm(tst,'E');
                   }
                   break;
               case 7:
                 mer_close();           /* sets joints to 0 position */
                 wn_exit();            /* close window stuff */
                 exit(0);
               case 8:
                 mer_close();         /* sets joints to 0 position */
                 mer_hshi_exit();      /* terminates the hshi pgm */
                 wn_exit();           /* close window stuff */
                 exit(0);
               default:
                 wn_exit();
                 exit(0);
                 break;
             }
      }
}
```

testsys.c

```
/*************************************************************

          FILENAME:      fullsys.c

          FUNCTION NAME:  full_system

          AUTHOR:        Todd Mosher & Monty Crabill

          DATE:          April 23 1991

     DESCRIPTION:
         This program will allow a user to control the merlin robot
         using the mba exo-skeleton.

*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>


#include "merlin.ref"
extern int stop_print;       /* this is a var in the tmwin.lib */

/* used to stop screen prints */ unsigned int utime();
extern float data[3][2000];
extern int datcnt;
extern int trigr;

float flimit(val)
float val;
{
    if (val > 1)
        return(1);
      if (val < -1)
        return(-1);
      return(val);
}

int test_system()
{
        long tm;
        char str[100];
        unsigned int ptime,ctime;
        int i,j,k,wptr;
        double x = 15;
        double y = 18;    /* y = 0 causes sqrt error --- fix for normal operation */
        double z = 7;
        double usedx = 0;
        double zforce;
        double lastx,lasty,lastz;
        long waste[6];
        double opinp[4];
        double resmat[4];
        FILE *fp[3];
        unsigned int cnts;
        double Ferror = 0;
        float Fdesired = 0;
        double Fscale = 0;
        double xd = 0;
        double prevx,prevy,prevz;
        float vel = 85;
        float Ftol = .01;
        float gain = 27;

     mba_rqst = 'L';
     j = 0;
     datcnt = 0;
     trigr = 0;
     clrscr();
/*   printf("\n\n        Loading Exo Program\n");
/*   system("upload mba.e68");                  /* load exo pgm */
/*   if ((fp[0] = fopen("upload.sta","r")) == NULL)
/*   {
/*     printf("Disk error or upload version error\n");
/*     printf("Press return to continue,  ctrl c to abort\n");
/*     getchar();
/*   }
/*   fscanf(fp[0],"%d",&i);        /* get result of upload to mba */
/*   fclose(fp[0]);
/*   if (i > 0)              /* upload problem */
/*     return(wind(8,8,"COMM ERR",'E',' ',"Can not communicate with MBA",NULL));
/*
/*   tty_open(2,10,8,1,0);        /* setup the serial port for mba */
/*
/*   tty_out(2,mba_rqst);           /* start handshaking with EXO */
/*
/*   if (i == 0)          /* Exo program newly loaded */
/*   {
/*     wind(8,8,"INITIALIZE",'E',' ',"Move all joints through their",
/*                            "Full range of motion",NULL);
/*     calib_mba();      /* Calibrate system */
/*   }
*/
     clrscr();
     fill_menu(25,8,"WRIST",  /* setup the menu */
                                "LOCK WRIST ON  ",  /* turn prints on/off for speed */
                                "LOCK WRIST OFF ",
                                NULL);
```

```
lock_wrist = !menu();
    fill_form(1,1,"Desired force ","  ",-1,-1,'F',6,&Fdesired);
    fill_form(1,3,"Desired Vel in/sec ","  ",-1,-1,'F',6,&vel);
    fill_form(1,5,"Force tolerance ","  ",-1,-1,'F',6,&Ftol);
    fill_form(1,7,"Gain (1-100)","  ",-1,-1,'F',6,&gain);
    form_disp(" Force control params ","  ",1,1,
                                    25,3,30,12); /* display initial form */
                   form_exe(&i,0);  /* get user input for xyz */
                   form_close();
                   xd = vel / 250.0;    /* 4 ms loop time = 250 */
                   if (gain == 0)
        gain = 1;
      Fscale = 100.0 / gain;



/*     stop_print = 1;  */
/*     while (!kbhit())      /* debug only */
/*     {
/*     outportb(0x300 + 4,0x00);
/*     outportb(0x300 + 4,0x02);
/*     }
/*getchar();
*/

    mba_init();       /* init vars etc */

    for (i=0;i<4;i++)     /* make sure hshi shows proper motor positions */
        mer_r_mpos(waste);

    tty_open(1,8,8,1,0);     /* setup the serial port for jr3 */

    tty_outs(1,"DP S\r");        /* do a clear buffer */
    tty_outs(1,"RO\r");       /* zero offsets */
    tm = time(NULL) + 2;
    while (time(NULL) < tm);
    while (tty_in(1) > 0);

    tty_outs(1,"EA = FZ\r"); /* use only z  this also starts hand shaking*/
    get_jr3_info(str,9);
    cnts = ((1.0/133.0) / .8380966e-6) * 2;
    wptr = status(1,1," INVERSE KINEMATICS INFO ",75,20);
    done = 0;
    sprintf(str,"xd = %6.3lf  Fd = %6.2f",xd,Fdesired);
    prints(0,str,1,5,0);
    while(!done)                /* until user aborts */
    {
outportb(0x304,00);      /* testing only!!! */
        prockey();
        prevx = x;
        prevy = y;
        prevz = z;
        lastx = mt6_0[0][3];        /* save prev xyz for indexing */
        lasty = mt6_0[1][3];        /* save prev xyz for indexing */
        lastz = mt6_0[2][3];        /* save prev xyz for indexing */

/*      joint_ang(wptr);            /* get joint angles from the mba */

/*      exot_mat();                 /* Computes all required matrix */

/*   elements */
outportb(0x304,02);       /* testing only !!! */
/*          if (indexing)                 /* set new indexing */
/*          {
/*               x_offset += lastx - mt6_0[0][3];
/*               y_offset += lasty - mt6_0[1][3];
/*               z_offset += lastz - mt6_0[2][3];
/*          }
/*          else                          /* calc new indexing position */
/*          {
/*               x = mt6_0[0][3] + x_offset; /* get x,y,and z -shift workspace */
/*               y = mt6_0[1][3] + y_offset;
/*               z = mt6_0[2][3] + z_offset;
/*          }
*/
        if (trigr)
        {
            if (datcnt == 0)
               ptime = utime();
             data[0][datcnt] = x;
             data[1][datcnt] = y;
             data[2][datcnt] = z;
             datcnt++;
             if (datcnt == 2000)
               trigr = 0;
             j = 0;
             while(1)              /* wait proper interval before collecting */
             {
               j++;
               ctime = utime();
               if (ptime - ctime >= cnts)   /* done waiting */
               break;                   /* go collect the data */
             }
             if (j < 3)           /* just not fast enough for the task */
               exit(printf("could not sample fast enough \n"
                               "ptime %u ctime %u\n",ptime,ctime));
                      ptime = ctime;
          }

        get_jr3_info(str,21);
```

```
            for (i=0;i<20;i++)
                 if (str[0] != 'F')
                       strcpy(str,&str[1]);
                    else
                       break;
                 if (i <20)    /* f found */
                 sscanf(&str[2],"%lf",&zforce);   /* get force from sensor */

            prints(0,str,50,1,0);
            sprintf(str,"%lf",zforce);
            prints(0,str,50,2,0);
/****************
                   force control loop
****************/

            Ferror = Fdesired + zforce;
            if ((Ferror < 0 && Ferror > -Ftol) ||
                   (Ferror > 0 && Ferror < Ftol))
                  Ferror = 0;

            x += flimit((float)(Ferror/Fscale)) * xd;

            sprintf(str,"x = %6.3lf  Fscale = %6.2lf limit %6.2lf err %6.2lf",
                                               x,Fscale,(double)(flimit((float)(Ferror/Fscale))),
                                               Ferror);
                             prints(0,str,1,6,0);
                             if (x < 12)              /* check mins and maxs */
                  x = 12;
               else if (x > 35.44)
                  x = 35.44;

/*       if (z < -23)                  /* do not allow crashing into floor */
/*        z = -23;
  */
            if (z < -28)                  /* do not allow crashing into floor */
                  z = -28;

                                               /* perform inverse kinematics for MERLIN */
                              if (sinvkin(&wptr,x,y,z,wrist_roll,wrist_flex,tool_roll) != 8)
                              {
                  x = prevx;
                  y = prevy;
                  z = prevz;
               }
         }

      if (datcnt)
      {
            fp[0] = fopen("xvout.dat","w");
            fp[1] = fopen("yvout.dat","w");
            fp[2] = fopen("zvout.dat","w");
```

testsys.c

```
    for (i=0;i<3;i++)
        fprintf(fp[i],"%c%s\nTime\nvel in/sec\n133\n",'X'+i,"vout");

    for (i=0;i<datcnt-1;i++)
        for(j=0;j<3;j++)
                                        /* calc velocity at 133 samples per sec */
                          data[j][i] = (data[j][i] - data[j][i+1]) / 7.5187699e-3;

    for (i=0;i<datcnt-1;i++)
    {
        fprintf(fp[0],"%f\n",data[0][i]);
        fprintf(fp[1],"%f\n",data[1][i]);
        fprintf(fp[2],"%f\n",data[2][i]);
      }
      fclose(fp[0]);
      fclose(fp[1]);
      fclose(fp[2]);
   }

  close_status();
  tty_flush(1);
  tty_flush(2);
  tty_close(2);
  tty_close(1);
  if (done == 2)
    wind(8,8,"TIMEOUT",'E',' ',"Timeout waiting for MBA joint angles",
                                                  NULL);
                        else if (done == 3)
    wind(8,8,"TIMEOUT",'E',' ',"Timeout waiting for JR3 data",
                                                  NULL);

}
```

getjr3.c

```
/***********************************************************

     FILENAME:       getjr3.c

          FUNCTION NAME:    get_jr3_info()

    AUTHOR:          Todd Mosher

     DATE:           07-15-91

    DESCRIPTION:
                     This routine will get the force and moment information
                     from the jr3 force torque sensor

***********************************************************/
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include "merlin.ref"

static char todd1[100] = {0};

int get_jr3_info(str,amt)
char *str;
{
        char tbuff[32];    /* temp input buffer */
        char byt;
        int j,i;
        char pstr[100];
        long tm;

    tm = time(NULL) + 2;          /* time out for data */ while(tty_cnt(JR3_PORT) < amt)  /*
    wait for all data to return from jr3 */
        if (time(NULL) >= tm)      /* timeout so getout */
        {
            if (kbhit() && getch() == 27)
            {
                done = 2;
                return;
              }
              tty_open(JR3_PORT,8,8,1,0);    /* setup the serial port for jr3 */
              sprintf(pstr,"%d Missing JR3 data",++badcnt);
              prints(0,pstr,25,1,0);
              while (tty_cnt(JR3_PORT))  /* clear the port */
                tty_in(JR3_PORT);
              tm = time(NULL) + 2;      /* setup time for new request */
              tty_outs(JR3_PORT,"DP S\r"); /* send new request for data */
          }
```

getjr3.c

```
for(j=0;j<amt;j++)                /* get the data */
        str[j] = tty_in(JR3_PORT);
      str[j] = NULL;

    if (tty_cnt(JR3_PORT))   /* bad condition - should not be any leftover */
    {                        /* charactors */
        for (;tty_cnt(JR3_PORT)>0;j++)
            str[j] = tty_in(JR3_PORT);
          str[j] = NULL;
          prints(0,str,10,10,0);
          prints(0,todd1,10,11,0);
      }
      strcpy(todd1,str);


    tty_outs(JR3_PORT,"DP S\r");     /* give next request to jr3 */ }
```

prockey.c

```
/************************************************************

        FILENAME:       prockey.c

        FUNCTION NAME:  prockey

        AUTHOR:         Todd Mosher

    DATE:           7-30-91

  DESCRIPTION:
      This will handle any operator input.

************************************************************/
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <math.h>

#include "merlin.ref"
extern int trigr,datcnt;

int prockey()
{
        int i;
        char str[100];
        long tm;

     if (!kbhit())     /* nothing to process */
        return;

     i = toupper(getch());   /* esc aborts this program */

     if (i == 27)
        done = 1;
       else if (i == 'S')        /* stops system until a q is pressed */ while(toupper(getch()) != 'Q');
       else if (i == 'T')       /* trigger to save data */
       {
         trigr = 1;
         datcnt = 0;
       }
       else if (i == 'R')
       {
         wind(8,8,"RELEASING LATCH",'I',' ',"Releasing latch to engage motors", NULL);
         tty_outs(1,"RL\r");        /* release latch */
         tm = time(NULL) + 2;
         while (time(NULL) < tm);
         while (tty_in(1) > 0);
         close_info();
```

prockey.c

```
tty_outs(1,"DP S\r");        /* restart data request */
     }
     else if (i == 'C')
     {
          wind(8,8,"CLR BUF",'I',' ',"Clearing jr3 serial buffers",
                                                            NULL);
                                      tm = time(NULL) + 2;
                                      while (time(NULL) < tm);
                                      while (tty_in(1) > 0);
                                      close_info();
                                      tty_outs(1,"DP S\r");        /* restart data request */
       }
       else if (i == 0)
          i = getch();
}
```

utime.c

```
/*****************************************

        This will return the number of counter
        tics in the 5253 counter timer.
        Each tic is approx .8380966us and the
        counter is reset every 53ms.
        This is a down counter.

*****************************************/
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

unsigned int utime()
{
        char arr[5];
        unsigned int *ptr;

        ptr = (unsigned int *)arr;

        outportb(0x43,0x00);      /* latch current count */
        arr[0] = inportb(0x40);   /* get data from counter */ arr[1] = inportb(0x40);
        return(ptr[0]);
}
```

keycont.c

```
/********************************************

    FILENAME:  keycont.c

    ROUTINE:   key_control

    AUTHOR:    TW Mosher

    DATE:      3-25-91

    DESCRIPTION:
        This routine will allow the user to set any joint to any
        position.  When done, this will be calibrated to the new
        zero position area.

**************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

int key_control(usrfun,ufparam)
int  (*usrfun)();          /* pass in desired function */
int *ufparam;              /* parameter for user function */
{
      char jnt[6][40] = { "base - left/right      ",
                           "shoulder - up/down     ",
                           "elbow - up/down        ",
                           "wrist roll - left/right",
                           "wrist flex - up/down   ",
                           "hand roll - left/right "
                           };
                  char str[100];
                  int i,joint = 0;
                  int k;
                  int wid;
                  float speed = .05;   /* slow */
                  double deg[6] = {0};

/*   mer_init_serv();    /* init servo values - gain ... */
     wind(23,0," ARROW CONTROL ",'I',' ',"1-6 To select joint",
                                          "F/S for fast/slow",
                                          "Use arrows to move ",
                                          "Press ESC when done",
                                          NULL);
                          wid = status(5,6," INFORMATION ",60,3);  /* put up a status
                          window */ prints(wid,"Control speed is slow",31,1,0);
                          prints(wid,jnt[joint],1,1,0);   /* print to status window */
                          while (1)              /* allow user to control the robot */
                          {
       if (usrfun)         /* if user function passed in, use it */
```

keycont.c

```
    usrfun(ufparam);     /* call user function */
if (kbhit())
{
    k = toupper(getch());          /* get the key */
    switch(k)              /* process the key */
    {
    case '1':
    case '2':
    case '3':                    /* joint selections */
    case '4':
    case '5':
    case '6':
            joint = k - '1';  /* convert to joint 0-5 */
            prints(wid,jnt[joint],1,1,0);   /* print to status window */ break;
        case 'F':
            prints(wid,"Control speed is fast",31,1,0);
            speed = 4;
            break;
        case 'S':
            prints(wid,"Control speed is slow",31,1,0);
            speed = .05;
            break;
        case 75:              /* left arrow */
            if (joint == 0 || joint == 3 || joint == 5)
                    deg[joint] -= speed;
                break;
        case 77:              /* right arrow */
            if (joint == 0 || joint == 3 || joint == 5)
                    deg[joint] += speed;
                break;
        case 72:              /* up arrow */
            if (joint == 2 || joint == 4)
                    deg[joint] -= speed;
                else if (joint == 1)
                    deg[joint] += speed;
                break;
        case 80:              /* down arrow */
            if (joint == 2 || joint == 4)
                    deg[joint] += speed;
                else if (joint == 1)
                    deg[joint] -= speed;
                break;
        case 27:      /* esc */
            close_info();
            close_status();
            return;
        default:
            break;
        }
}
```

```
            if (deg[joint] > degmax[joint])
                deg[joint] = degmax[joint];
              else if (deg[joint] < degmin[joint])
                deg[joint] = degmin[joint];

            movearm(deg,'D');    /* move to new encoder position */
          }
}
```

calibmer.c

```
/********************************************

     FILENAME:   calibmer.c

     ROUTINE:    calib_mer

     AUTHOR:     TW Mosher

     DATE:       3-25-91

     DESCRIPTION:
          This routine will allow the user to set any joint to any position.  When done, this will be
          calibrated to the new zero position area.

****************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

int calib_mer()
{
        char str[100];

     mer_init_serv();    /* init servo values - gain ... */ key_control(NULL,NULL);
     mer_init_serv();
}
```

movearm.c

```
/*********************************************

    FILENAME:    movearm.c

    PROGRAM:     movearm

    AUTHOR:      TW Mosher

  DATE:       3-21-91

    DESCRIPTION:
        This will command the motors to the proper position
        which was passed in.  The passed in vals can be radians,
        degrees, or encoder counts.

    Parameters
        pointer to an array of 6 rad or deg  (double) or encoder (long)
        char  r for rad, d for deg, e for encoder
      note - no range checking done here.
      Returns 0 on success, else non 0 = error
                                       -1 = invalid data type passed in
                                       -2 = can not get control from hshi
**************************************************/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include "merlin.ref"

static long lastpos[6];

int movearm(data,typ)
double *data;
char typ;
{
        int i;
        long *ldata;
        double dtmp;
        long encod[6] = {0};
        char str[100];
        char str1[100];
        int maxmov;

    ldata = (long *)data;    /* make either type the correct addr */
    typ = toupper(typ);      /* ensure no typo problem */
    if (typ == 'D')          /* degrees to encoders */
    {
        encod[0] = data[0] * dtoe[0];   /* convert joint 0 */
        encod[1] = data[1] * dtoe[1];   /* convert joint 1 */
        encod[2] = (data[2] - data[1]) * dtoe[2];  /* convert joint 2 */
        encod[3] = data[3] * dtoe[3];   /* convert joint 3 */
```

```
                                                  /* joints 3,4,&5 are coupled together */
                                  dtmp = data[4] * DTORAD * 1.2; /* 1.2 is ratio hshi manual */
                                  encod[4] = (data[3] * DTORAD - dtmp) * 7639.437;
                                                                /* 7639.437 is */
                                                                /* ticks / rad hshi manual */
                                         encod[5] = (((data[3] - data[5]) * DTORAD) + dtmp) * 7639.437;
   }
   else if (typ == 'R')
   {
     encod[0] = data[0] * rtoe[0];   /* convert joint 0 */
     encod[1] = data[1] * rtoe[1];   /* convert joint 1 */
                                encod[2] = (data[2] - data[1]) * rtoe[2];
                                                                /* convert joint 2 */
                                         encod[3] = data[3] * rtoe[3];   /* convert joint 3 */
                                                  /* joints 3,4,&5 are coupled together */
                                  dtmp = data[4] * 1.2;   /* 1.2 is ratio from hshi manual */
                                  encod[4] = (data[3] - dtmp) * 7639.437;     /* 7639.437 is */
                                                             /* ticks / rad from hshi manual */
                                       encod[5] = (data[3] - data[5] + dtmp) * 7639.437;
   }
   else if (typ == 'E')
     for (i=0;i<6;i++)
         encod[i] = ldata[i];
     else
     return(-1);        /* bad type passed in */

                                      /* the motors can attempt a 30 deg increment */
                                      /* when the motors are on, but can only attempt */
                                      /* a 1 degree increment when the motors are off */

maxmov = 100;  /* assume motors off - can not move - .5 deg max */
for (i=0;i<6;i++)                    /* save prev val */
     if (lastpos[i] != hr_mpos->axis[i]) /* has moved - motors ok */
     {
         maxmov = 8000;  /* while tracking - 30 deg max increments */
         break;
       }

for (i=0;i<6;i++)                   /* save prev motor position */
     lastpos[i] = hr_mpos->axis[i];
  for (i=0;i<6;i++)     /* max motion per frame = 30 deg */
  {
     if (encod[i] > hr_mpos->axis[i] + maxmov)
         encod[i] = hr_mpos->axis[i] + maxmov;  /* limit max motion */
       else if (encod[i] < hr_mpos->axis[i] - maxmov)
         encod[i] = hr_mpos->axis[i] - maxmov;
     }

mer_get_ctrl();  /* do not update window until we are in control */
str[0] = NULL;
str1[0] = NULL;
```

movearm.c

```
for (i=0;i<6;i++)    /* update merlin joint areas */
     {
        hc_mpos->axis[i] = encod[i];
        sprintf(&str[strlen(str)]," %6ld",encod[i]);
        sprintf(&str1[strlen(str1)]," %6ld",hr_mpos->axis[i]);
      }
      prints(0,str,1,10,0);
      prints(0,str1,1,11,0);

     mer_mov();                    /* move merlin to new position */
}
```

mercmds.c

```
/*********************************************

    FILENAME:  mercmds.c

    ROUTINE:   several low level cmds to control the merlin

    AUTHOR:    TW Mosher

    DATE:      3-25-91

    DESCRIPTION:
        mer_init_serv  - does a set servo parameters command - this both
                         sets the servo stuff and resets the joint computers
                mer_mov     - sets the merlin command to move. - joint values
                         assumed to be correct at this time
                mer_cmd     - does actual handshaking with the merlin when setting commands.
                mer_r_mpos  - reads motor position encoders from window
                mer_close   - resets joints to 0 position
                mer_hshi_exit - exits hshi pgm

*****************************************************/
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include "merlin.ref"

int mer_init_serv()
{
        int i;

    for (i=0;i<6;i++)    /* set all servo parameters */
    {
        hc_srv->servo_param[i].max_acc = max_srv_acc[i]; hc_srv->servo_param[i].max_vel =
        max_srv_vel[i]; hc_srv->servo_param[i].gain = gain[i];
      }
      mer_cmd(CMD_SET_PARAMS);    /* have merlin do command */
}

int mer_r_mpos(mp_encoders)
long *mp_encoders;
{
        int i;
      mer_cmd(CMD_RD_M_STAT);        /* this forces a status update */
      for (i=0;i<6;i++)              /* read all motor position encoders */ mp_encoders[i] =
        hr_mpos->axis[i];
}
```

mercmds.c

```
int mer_r_jpos(mp_rad)
double *mp_rad;
{
        int i;
       mer_cmd(CMD_RD_J_STAT);        /* this forces a status update */
       mer_cmd(CMD_RD_J_STAT);        /* this forces a status update */
       for (i=0;i<6;i++)             /* read all joint angles */
          mp_rad[i] = hr_jpos->axis[i];
}

int mer_mov()                /* simple move command -- all joints */
{                         /* have already been set */ mer_cmd(CMD_M_POS);
}

int mer_get_ctrl()
{
        long tm;

     tm = time(NULL) + 2;              /* max wait for merlin robot */ while(hc_buf->buf_stat !=
     BUF_HOST)  /* wait to gain control */
          if (time(NULL) > tm)        /* only occasionally check for keystroke */ {
             if (kbhit() && getch() == 27)     /* of hshi -- allow user abort */ break;
             tm  = time(NULL) + 2;
             printf(" Merlin is not Powered up properly\n");
            }
}

int mer_cmd(cmd)
int cmd;
{
     mer_get_ctrl();      /* make sure we are in control of buffer */
     hc_buf->command = cmd;   /* set the command word */
     hc_buf->buf_stat = BUF_HSHI;  /* give merlin control */
}

int  mer_close()      /* set robot to zero position */
{
        int i;
        int j;
        double angs[6];

     mer_r_jpos(angs);        /* get current position */
     for (i=0;i<6;i++)
     {
        hc_srv->servo_param[i].max_acc = 1;    /* set all servos to min */ hc_srv-
        >servo_param[i].max_vel = 1;
        hc_srv->servo_param[i].gain = 2;
      }
```

```
    mer_cmd(CMD_SET_PARAMS);    /* set servos - this inits joint position */ for (i=0;i<6;i++)
        hc_jpos->axis[i] = -angs[i];/* move from new 0 to prev angles */
      mer_cmd(CMD_J_POS);            /* have merlin move slow */ wind(8,8,"WAIT",'E','
      ',"Press return when MERLIN",
                                        "Has returned to its origin",
                                        NULL);
}

int mer_hshi_exit()
{
    mer_cmd(CMD_EXIT);
}
```

window.c

```
/*********************************************************************** *** NAME OF
            ROUTINE: window_init
            NAME OF FILE:   window
            CREATION DATE:  05/22/89
            AUTHOR:         T. Mosher

            DESCRIPTION:
               This will initialize the 32k dual port ram that
               is in the VME chasis.

     REVISIONS
   REV#     DESCRIPTION                            INITIALS  DATE
   ---   ----------------------------------   ---   --------

*********************************************************************** **/

/*#define MEM_OFFSET  0x0d0000000  */
#include "merlin.ref"
#define ATCMD 0x0200

int window_init()
{
        int i;
        int at_cmd = ATCMD;
        int at_stat = ATCMD + 2;
        int vme_stat = ATCMD + 8;
        int vme_am = ATCMD + 13;
        int stat;

     inportb(at_stat);
     inportb(vme_stat);
     stat = inportb(at_stat);     /* b added by steve */
     if (stat & 1)
     {
         printf("Power is off or cable is disconnected.\n");
         return(0);
       }
       outportb(at_cmd,128);
       outportb(vme_am,61);
       stat = inportb(at_stat);   /* b added by steve */
       if (stat & 197)
       {
         printf("Setup status error:\n");
         if (stat & 128)
              printf("interface parity error\n");
           if (stat & 64)
              printf("vme bus error\n");
           if (stat & 4)
              printf("interface timeout\n");
           if (stat & 1)
```

```
            printf("power is off or cable is disconnected\n");
          return(0);
      }
      hc_buf->buf_stat = BUF_HOST;    /* gain control of window from merlin */ return(1);
}
```

forkin.c

```
/**********************************************************

    FILENAME:      forkin.c

    FUNCTION NAME:  forkin()

    AUTHOR:        Mosher & Crabill

    DATE:         22 March 91

    MODIFIED:      3 April 91

   DESCRIPTION:    This program will allow checkout of the Merlin
                       left arm forward kinematics.

**********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "merlin.ref"

int forkin(wid)
int *wid;
{
        int i;
        int wp;
        double resmat[4];
        long motor_encoder[6];
        char str[100];

    wp = *wid;     /* get window id */

    mer_r_mpos(motor_encoder);     /* read motor position encoders */
    prints(wp,"Motor encoder values",14,0,1);
    str[0] = 0;
    for(i=0;i<6;i++)
        sprintf(&str[strlen(str)],"%6ld ",motor_encoder[i]);
      prints(wp,str,1,1,0);
      enc_to_rad(motor_encoder,in_merlin_joint,"MERLIN");
/**********************/

    in_merlin_joint[2] -= 1.5707963;  /* -90 deg correction for elbow */

/**********************/

    merltmat();                  /* Merlin's tool roll/global reference */
                                               /* transformation matrix elements */
                         prints(wp,"Joint angles",18,2,1);
```

forkin.c

```
str[0] = 0;
     for (i=0;i<6;i++)
          sprintf(&str[strlen(str)],"%6.2lf ",in_merlin_joint[i] * 57.29578); /* display degrees */
        prints(wp,str,1,3,0);

     matmlt(st6_0,gripper_tip,resmat,wp);
                                                       /* multiply tool roll/ global ref */
                                                       /* transformation matrix by gripper */
                                                       /* tip position matrix */
/*   }
*/
}
```

sinvkin.c

```
/*************************************************************

  FILENAME:      sinvkin.c    smart inverse kinematics

    FUNCTION NAME:  sinvkin()

    AUTHOR:         Mosher & Crabill

    DATE:         09 March 91

    MODIFIED:

   DESCRIPTION:
        This program will take an xyz position and use the
        3 main axis of motion to move the wrist to the position.
        This then calls the procedure to determin the wrist angles
        and then finally does the actual moveing of the robot to
        the correct place.

*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "merlin.ref"
/******************************
*
*   Calculate the k and g function values
*
******************************/
double skgfun(indx,typ)
int indx;
char typ;   /* g or k */
{
        double f[4],k[5],g[5];
        double ct3,ca2;

    typ = toupper(typ);

    ct3 = cos(theta[3]);
    ca2 = cos(alpha[2]);
    f[1] = a[3] * ct3 + d[4] * sin(alpha[3]) * sin(theta[3]) + a[2];
    f[2] = a[3] * sin(theta[3]) * ca2 -
            d[4] * sin(alpha[3]) * ct3 * ca2 -
            d[4] * cos(alpha[3]) * sin(alpha[2]) -
            sin(alpha[2]) * d[3];
        f[3] = a[3] * sin(theta[3]) * sin(alpha[2]) -
            d[4] * sin(alpha[3]) * ct3 * sin(alpha[2]) +
            d[4] * cos(alpha[3]) * ca2 +
            ca2 * d[3];
        if (typ != 'G')   /* must be a k function request */
```

sinvkin.c

```
{
        k[1] = f[1];
        k[2] = -f[2];
        k[3] = sqr(f[1]) + sqr(f[2]) + sqr(f[3]) + sqr(a[1]) + sqr(d[2]) +
                       2 * d[2] * f[3];
               k[4] = f[3] * cos(alpha[1]) + d[2] * cos(alpha[1]);
               return(k[indx]);
      }
      else
      {
        g[1] = f[1] * cos(theta[2]) - f[2] * sin(theta[2]) + a[1];
        g[2] = f[1] * sin(theta[2]) * cos(alpha[1]) +
                 f[2] * cos(theta[2]) * cos(alpha[1]) - f[3] * sin(alpha[1]) -sin(alpha[1]) * d[2];
             return(g[indx]);
      }
}

/****************************
*
*     Limit some motions
*
****************************/
double limit(val)     /* only for ranges from -pi/2 to pi/2 */
double val;
{
    if (val <= -1.5 * PI)
        return(2 * PI + val);
      if (val >= 1.5 * PI)
        return(-2 * PI + val);
      return(val);
}


int sinvkin(wid,x,y,z,wrist_roll,wrist_flex,tool_roll)
int *wid;
double x,y,z,wrist_roll,wrist_flex,tool_roll;
{
      double r,f[4],k[5],g[5];
      char str[100];
      double thetat[2];     /* temp theta */
      int i;
      int wptr;

    wptr = *wid;

/*  sprintf(str,"x,y,z = %lf %lf %lf",x,y,z);
/*  prints(0,str,0,0,0);
/*  getchar();
    */
    r = sqr(x) + sqr(y) + sqr(z);
    if (sqrt(r) < 18.5)
        return(prints(wptr,"TOO CLOSE",0,0,1));
      else if (sqrt(r) > 36)
        return(prints(wptr,"TOO FAR  ",0,0,1));
/*    prints(wptr,"         ",0,0,0);*/
    h[1] = r + h1_partial;
         /************** theta  3   **************/
      thetat[0] = 2 * atan2((double)(2 * h[3] +
                      sqrt((double)(4 * sqr(h[3]) - 4 * (sqr(h[1])-sqr(h[2]))))), (double)(2 * (h[1] +
                      h[2])));
          thetat[1] = 2 * atan2((double)(2 * h[3] -
                       sqrt((double)(4 * sqr(h[3]) - 4 * (sqr(h[1])-sqr(h[2]))))), (double)(2 * (h[1]
                       + h[2])));

    if (thetat[1] <= -PI/2.0)       /* elbow up */
        theta[3] = thetat[1];
      else
        theta[3] = thetat[0];
/*    sprintf(str,"theta 3 %8.2lf",(double)(theta[3]*57.295));*/
      /************* theta   2   ***************/
   for (i=1;i<5;i++)
    k[i] = skgfun(i,'K');     /* get k value */
   thetat[0] = 2 * atan2((double)(-sin(alpha[1]) * k[1] +
           sqrt((double)(sqr(sin(alpha[1])) * sqr(k[1]) -
            (sqr(z) + 2 * z * k[4] - sqr(sin(alpha[1])) *
            sqr(k[2]) + sqr(k[4])))),
            (double)(z + k[4] + sin(alpha[1]) * k[2]));

    thetat[1] = 2 * atan2((double)(-sin(alpha[1]) * k[1] -sqrt((double)(sqr(sin(alpha[1])) * sqr(k[1]) -
                      (sqr(z) + 2 * z * k[4] - sqr(sin(alpha[1])) * sqr(k[2]) + sqr(k[4])))),
                      (double)(z + k[4] + sin(alpha[1]) * k[2]));

    thetat[0] = limit(thetat[0]);
    if (thetat[0] >= PI / 2.0 && thetat[0] <= -PI / 2.0)
        theta[2] = thetat[0];
      else
        theta[2] = limit(thetat[1]);

/*    sprintf(str,"theta 2 %8.2lf",(double)(theta[2]*57.295));*/
         /************* theta    1   ***************/
       for (i=1;i<3;i++)
       g[i] = skgfun(i,'G');
      thetat[0] = 2 * atan2((double)(-(x * g[1] - y * g[2]) + sqrt((double)((sqr(x) + sqr(y)) *
                      (sqr(g[1]) + sqr(g[2]))))), (double)(-(y * g[1] + x * g[2])));

    thetat[1] = 2 * atan2((double)(-(x * g[1] - y * g[2]) -sqrt((double)((sqr(x) + sqr(y)) * (sqr(g[1])
                      +
                    sqr(g[2]))))), (double)(-(y * g[1] + x * g[2])));

    thetat[0] = limit(thetat[0]);

    if (thetat[0] <= PI / 2.0 && thetat[0] >= -PI / 2.0)
        theta[1] = thetat[0];
      else
        theta[1] = limit(thetat[1]);

/*    sprintf(str,"theta 1 %8.2lf",(double)(theta[1]*57.295));  */

    wrist_angles();          /* calculate euler angles for MERLIN wrist */

    theta[3] += PI / 2.0;      /* correct 90 deg for lo level drivers */

    for (i=0;i<3;i++)          /* correct for indexing */
        theta[i] = theta[i+1];

    if (wrist_flex > 1.4835)    /* limit wrist_flex to 85 deg */
        wrist_flex = 1.4835;
      else if (wrist_flex < -1.4835)
        wrist_flex = -1.4835;
      theta[3] = wrist_roll;
      theta[4] = wrist_flex;
      theta[5] = tool_roll;

    movearm(theta,'R');         /* move the robot to the desired positions */ return(8);
}
```

merinit.c

```
/************************************************************

      FILENAME:        merinit.c

            FUNCTION NAME:    merinit

            AUTHOR:          TW Mosher ML Crabill

      DATE:           4/9/91

            MODIFIED:

   DESCRIPTION:
               Assorted initializations for variables.

************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "merlin.ref"

int merinit()
{
               /****************************
               *
               *   init section for inverse kin vars
               *
               ****************************/
    h1_partial = -sqr(a[3]) - sqr(a[2]) - sqr(a[1]) - sqr(d[4]) -sqr(d[3]) -sqr(d[2]) - 2 * d[4] * d[3] *
                         cos(alpha[3]) -
                         2 * d[2] * (d[4] * cos(alpha[3]) * cos(alpha[2]) + cos(alpha[2]) * d[3]);
    h[2] = 2 * a[2] * a[3] - 2 * d[2] * d[4] * sin(alpha[3]) * sin(alpha[2]); h[3] = 2 * a[2] * d[4] *
    sin(alpha[3]) + 2 * d[2] * a[3] * sin(alpha[3]); }
```

merltmat.c

```
/*************************************************************
    FILENAME:       merltmat.c

    FUNCTION NAME:  merltmat()

    AUTHOR:         Mosher & Crabill

  DATE:         03/22/91

    MODIFIED:       04/02/91
                         11:00 am

    DESCRIPTION:    Forward kinematic transformation matrices
                         for Merlin left armed robot.

*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

merltmat()
{
double ct1,nct1,st1,nst1,ct2,nct2,st2,nst2,ct3,nct3,st3,nst3,
         ct4,nct4,st4,nst4,ct5,nct5,st5,nst5,ct6,nct6,st6,nst6;

/*  Merlin Left Arm Robot Matrix[1] to transform from coordinate system at Waist/Shoulder
       Intersect back to Reference. */

    ct1 = cos(in_merlin_joint[0]);
    nct1 = -ct1;
    st1 =  sin(in_merlin_joint[0]);
    nst1 = -st1;

    st1_0[0][0] = ct1;
    st1_0[0][1] = nst1;
    st1_0[0][2] = 0;
    st1_0[0][3] = 0;
    st1_0[1][0] = nst1;
    st1_0[1][1] = nct1;
    st1_0[1][2] = 0;
    st1_0[1][3] = 0;
    st1_0[2][0] = 0;
    st1_0[2][1] = 0;
    st1_0[2][2] = -1;
    st1_0[2][3] = 0;

/*  Merlin Left Arm Robot Matrix[2] to transform from coordinate system at Shoulder back to
       Reference. */

    ct2 = cos(in_merlin_joint[1]);
    nct2 = -ct2;
    st2 = sin(in_merlin_joint[1]);
    nst2 = -st2;

    st2_0[0][0] = st1_0[0][0] * ct2;
    st2_0[0][1] = st1_0[0][0] * nst2;
    st2_0[0][2] = st1_0[0][1];
    st2_0[0][3] = st1_0[0][1] * -12.00;
    st2_0[1][0] = st1_0[1][0] * ct2;
    st2_0[1][1] = st1_0[1][0] * nst2;
    st2_0[1][2] = st1_0[1][1];
    st2_0[1][3] = st1_0[1][1] * -12.00;
    st2_0[2][0] = st1_0[2][2] * nst2;
    st2_0[2][1] = st1_0[2][2] * nct2;
    st2_0[2][2] = 0.0;
    st2_0[2][3] = 0.0;

/*  Merlin Left Arm Robot Matrix[3] to transform from coordinate system at Elbow/Wrist Roll
       intersect back to Reference. */

    ct3 = cos(in_merlin_joint[2]);
    nct3 = -ct3;
    st3 = sin(in_merlin_joint[2]);
    nst3 = -st3;

    st3_0[0][0] = st2_0[0][0] * ct3 + st2_0[0][1] * nst3;
    st3_0[0][1] = st2_0[0][0] * nst3 + st2_0[0][1] * nct3;
    st3_0[0][2] = -st2_0[0][2];
    st3_0[0][3] = st2_0[0][0] * 17.3 + st2_0[0][3];
    st3_0[1][0] = st2_0[1][0] * ct3 + st2_0[1][1] * nst3;
    st3_0[1][1] = st2_0[1][0] * nst3 + st2_0[1][1] * nct3;
    st3_0[1][2] = -st2_0[1][2];
    st3_0[1][3] = st2_0[1][0] * 17.3 + st2_0[1][3];
    st3_0[2][0] = st2_0[2][0] * ct3 + st2_0[2][1] * nst3;
    st3_0[2][1] = st2_0[2][0] * nst3 + st2_0[2][1] * nct3;
    st3_0[2][2] = -st2_0[2][2];
    st3_0[2][3] = st2_0[2][0] * 17.3 + st2_0[2][3];

/*  Merlin Left Arm Robot Matrix[4] to transform from coordinate system at Wrist Roll/Flex
       intersect back to Reference. */

    ct4 = cos(in_merlin_joint[3]);
    nct4 = -ct4;
    st4 = sin(in_merlin_joint[3]);
    nst4 = -st4;

    st4_0[0][0] = st3_0[0][0] * ct4 + st3_0[0][2] * nst4;
    st4_0[0][1] = st3_0[0][0] * nst4 + st3_0[0][2] * nct4;
    st4_0[0][2] = st3_0[0][1];
    st4_0[0][3] = st3_0[0][1] * 17.25 + st3_0[0][3];
    st4_0[1][0] = st3_0[1][0] * ct4 + st3_0[1][2] * nst4;
    st4_0[1][1] = st3_0[1][0] * nst4 + st3_0[1][2] * nct4;
    st4_0[1][2] = st3_0[1][1];
    st4_0[1][3] = st3_0[1][1] * 17.25 + st3_0[1][3];
    st4_0[2][0] = st3_0[2][0] * ct4 + st3_0[2][2] * nst4;
    st4_0[2][1] = st3_0[2][0] * nst4 + st3_0[2][2] * nct4;
    st4_0[2][2] = st3_0[2][1];
    st4_0[2][3] = st3_0[2][1] * 17.25 + st3_0[2][3];

/*  Merlin Left Arm Robot Matrix[5] to transform from coordinate system at Wrist Roll/Flex
       intersect back to Reference. */

    ct5 = cos(in_merlin_joint[4]);
    nct5 = -ct5;
    st5 = sin(in_merlin_joint[4]);
    nst5 = -st5;

    st5_0[0][0] = st4_0[0][0] * ct5 + st4_0[0][2] * nst5;
    st5_0[0][1] = st4_0[0][0] * nst5 + st4_0[0][2] * nct5;
    st5_0[0][2] = st4_0[0][1];
    st5_0[0][3] = st4_0[0][3];
    st5_0[1][0] = st4_0[1][0] * ct5 + st4_0[1][2] * nst5;
    st5_0[1][1] = st4_0[1][0] * nst5 + st4_0[1][2] * nct5;
    st5_0[1][2] = st4_0[1][1];
    st5_0[1][3] = st4_0[1][3];
    st5_0[2][0] = st4_0[2][0] * ct5 + st4_0[2][2] * nst5;
    st5_0[2][1] = st4_0[2][0] * nst5 + st4_0[2][2] * nct5;
    st5_0[2][2] = st4_0[2][1];
    st5_0[2][3] = st4_0[2][3];

/*  Merlin Left Arm Robot Matrix[6] to transform from coordinate system at Wrist Flex/Tool Roll
       intersect back to Reference. */

    ct6 = cos(in_merlin_joint[5]);
    nct6 = -ct6;
    st6 = sin(in_merlin_joint[5]);
    nst6 = -st6;

    st6_0[0][0] = st5_0[0][0] * ct6 + st5_0[0][2] * nst6;
    st6_0[0][1] = st5_0[0][0] * nst6 + st5_0[0][2] * nct6;
    st6_0[0][2] = st5_0[0][1];
    st6_0[0][3] = st5_0[0][3];
    st6_0[1][0] = st5_0[1][0] * ct6 + st5_0[1][2] * nst6;
    st6_0[1][1] = st5_0[1][0] * nst6 + st5_0[1][2] * nct6;
    st6_0[1][2] = st5_0[1][1];
    st6_0[1][3] = st5_0[1][3];
    st6_0[2][0] = st5_0[2][0] * ct6 + st5_0[2][2] * nst6;
    st6_0[2][1] = st5_0[2][0] * nst5 + st5_0[2][2] * nct6;
    st6_0[2][2] = st5_0[2][1];
    st6_0[2][3] = st5_0[2][3];
}
```

utils.c

```
/*********************************************************

  FILENAME:        util.c

    FUNCTION NAME:    several useful routines

    AUTHOR:          Todd Mosher

  DATE:           12-27-90

    DESCRIPTION:


*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

int matmlt(tmat,inmat,resmat)
double tmat[4][4];
double *inmat;
double *resmat;
{
        register int i,j;
        char str[100];

/*      prints("T MATRIX INFO",17,5,1);
     str[0] = 0;
     for (i=0;i<4;i++)
     {
          str[0] = 0;
          for(j=0;j<4;j++)
               sprintf(&str[strlen(str)],"%10.5lf ",tmat[i][j]); prints(str,1,6 + i,0);
          }

     prints("GRIPPER TIP",51,5,1);
*/

     for (i=0;i<4;i++)
     {
          resmat[i] = 0;
          for(j=0;j<4;j++)
               resmat[i] += tmat[i][j] * inmat[j]; sprintf(str,"%10.5lf ",resmat[i]);
/*             prints(wp,str,50,6+i,0);
*/
     }
}
int matmlt33(tmat,inmat,resmat)
```

utils.c

```
double tmat[4][4];
double inmat[4][4];
double resmat[4][4];
{
          register int i,j,k;

      for (i=0;i<3;i++)
      {
            for (j=0;j<3;j++)
                  resmat[j][i] = 0;
               for (j=0;j<3;j++)
                  for (k=0;k<3;k++)
                        resmat[j][i] += tmat[j][k] * inmat[k][i];
               }
}


matdisp(wptr,basy,dispmat)
double dispmat[4][4];
int basy;
{
          register int i,j,k;
          char str[100];

      for (i=0;i<3;i++)
      {
            str[0] = NULL;
            for(j=0;j<4;j++)
                  sprintf(&str[strlen(str)]," %7.2lf ",dispmat[i][j]); prints(wptr,str,1,i+basy,0);

      }
}
```

enctorad.c

```
/*******************************************

    FILENAME:    enctorad.c

    PROGRAM:     enc_to_rad

    AUTHOR:      Monty Crabill

  DATE:       4-3-91

    DESCRIPTION:

    Parameters
*************************************************/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include "merlin.ref"

int enc_to_rad(enc_in,rad_out,dev)
long *enc_in;
double *rad_out;
char *dev;          /* valid dev = MBA or MERLIN */
{
        double dtmp;

    if (toupper(dev[1]) == 'E')
    {
        rad_out[0] = enc_in[0] / rtoe[0];   /* convert joint 0 */
        rad_out[1] = enc_in[1] / rtoe[1];   /* convert joint 1 */
        rad_out[2] = enc_in[2] / rtoe[2] + rad_out[1];   /* convert joint 2 */
        rad_out[3] = enc_in[3] / rtoe[3];   /* convert joint 3 */
                                           /* joints 3,4,&5 are coupled together */
                        dtmp = enc_in[4] / rtoe[4];
                        rad_out[4] = (rad_out[3] - dtmp) * 0.8333;
                        rad_out[5] = (2 * rad_out[3] - dtmp - (enc_in[5] / rtoe[5]));
     }
}
```

fullsys.c

```
/*************************************************************

        FILENAME:       fullsys.c

        FUNCTION NAME:  full_system

        AUTHOR:         Todd Mosher & Monsy Crabill

        DATE:           April 23 1991

    DESCRIPTION:
        This program will allow a user to control the merlin robot
        using the mba exo-skeleton.

*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "merlin.ref"
extern int stop_print;      /* this is a var in the tmwin.lib */

/* used to stop screen prints */ unsigned int utime();
float data[3][2000];
int datcnt = 0;
int trigr = 0;

full_system()
{
        long tm;
        char str[100];
        unsigned int ptime,ctime;
        int i,j,k,wptr;
        double x = 28;
        double y = 18;   /* y = 0 causes sqrt error --- fix for normal operation */
        double z = 7;
        double zforce;
        double lastx,lasty,lastz;
        long waste[6];
        unsigned int cnts;
        double opinp[4];
        double resmat[4];
        FILE *fp[3];

    badcnt = 0;        /* count of bad comm packets */
    fill_menu(25,8,"ARM?", /* setup menu for selecting arm to use */
                                "Right arm ",
                                "Left  arm ",
                                NULL);
    if (!menu())            /* right arm selected */
        mba_rqst = 'R';
      else                  /* left arm selected */
        mba_rqst = 'L';


    j = 0;
    clrscr();
    printf("\n\n       Loading Exo Program\n");
    system("upload mba.e68");                /* load exo pgm */

    if ((fp[0] = fopen("upload.sta","r")) == NULL)
    {
        printf("Disk error or upload version error\n");
        printf("Press return to continue,  ctrl c to abort\n");
        getchar();
      }
      fscanf(fp[0],"%d",&i);       /* get result of upload to mba */
      fclose(fp[0]);
      if (i > 0)             /* upload problem */
        return(wind(8,8,"COMM ERR",'E',' ',"Can not communicate with MBA",NULL));

    tty_open(MBA_PORT,10,8,1,0);    /* setup the serial port for mba */

    tty_out(MBA_PORT,mba_rqst);     /* start handshaking with EXO */

    clrscr();
    wptr = status(1,1," INVERSE KINEMATICS INFO ",75,20);

    if (i == 0)          /* Exo program newly loaded */
    {
        wind(8,8,"INITIALIZE",'E',' ',"Move all joints through their",
                                                  "Full range of motion",NULL);
                              calib_mba();      /* Calibrate system */
      }

    fill_menu(25,8,"WRIST",  /* setup the menu */
                                    "LOCK WRIST ON  ",  /* turn prints on/off for speed */
                                    "LOCK WRIST OFF ",
                                    NULL);
                  lock_wrist = !menu();


/*     stop_print = 1;  */
/*     while (!kbhit())       /* debug only */
/*     {
/*     outportb(0x300 + 4,0x00);
/*     outportb(0x300 + 4,0x02);
/*     }
/*getchar(); */
```

```
    mba_init();      /* init vars etc */

    for (i=0;i<4;i++) /* make sure hshi shows proper motor positions */
        mer_r_mpos(waste);

    tty_open(JR3_PORT,8,8,1,0);    /* setup the serial port for jr3 */

    tty_outs(JR3_PORT,"DP S\r");        /* do a clear buffer */
    tty_outs(JR3_PORT,"RO\r");        /* zero offsets */
    tm = time(NULL) + 2;
    while (time(NULL) < tm);        /* wait for 2 seconds */
    while (tty_in(JR3_PORT) > 0);    /* now clear the buffers */

    tty_outs(JR3_PORT,"EA = FZ\r"); /* use only z  this starts hand shaking*/
    get_jr3_info(str,9);
    cnts = ((1.0/133.0) / .8380966e-6) * 2;
    done = 0;



    while(!done)                /* until user aborts */
    {
outportb(0x304,00);      /* testing only!!! */
        prockey();    /* processes s (stop) q(continue) t(trigger) esc(exit) */

        lastx = mt6_0[0][3];        /* save prev xyz for indexing */
        lasty = mt6_0[1][3];        /* save prev xyz for indexing */
        lastz = mt6_0[2][3];        /* save prev xyz for indexing */

        joint_ang(wptr);            /* get joint angles from the mba */

        r_exo_tmat();               /* Computes all required matrix */
                                               /*   elements */
                            matdisp(0,1,mt6_0);
outportb(0x304,02);       /* testing only !!! */
        if (indexing)               /* set new indexing */
        {
            x_offset += lastx - mt6_0[0][3];
            y_offset += lasty - mt6_0[1][3];
            z_offset += lastz - mt6_0[2][3];
          }
          else                          /* calc new indexing position */
          {
            x = mt6_0[0][3] + x_offset; /* get x,y,and z - shift workspace */
            y = mt6_0[1][3] + y_offset;
            z = mt6_0[2][3] + z_offset;
          }
          sprintf(str,"xyz = %7.2lf  %7.2lf  %7.2lf",x,y,z);
```

```
prints(0,str,45,3,0);

        if (trigr)   /* set by prockey - used to store xyz info into a file */
        {            /* press t to start the trigger */
            if (datcnt == 0)
                ptime = utime();
              data[0][datcnt] = x;
              data[1][datcnt] = y;
              data[2][datcnt] = z;
              datcnt++;
              if (datcnt == 2000)    /* collect only 2000 samples */
                trigr = 0;
              j = 0;
              while(1)       /* wait proper interval before collecting */
              {
                j++;
                ctime = utime();
                if (ptime - ctime >= cnts)   /* done waiting */
                break;                      /* go collect the data */
              }
              if (j < 3)          /* just not fast enough for the task */
                exit(printf("could not sample fast enough \n"
                                "ptime %u ctime %u\n",ptime,ctime));
                      ptime = ctime;
          }

        get_jr3_info(str,21);
        for (i=0;i<20;i++)
            if (str[0] != 'F')
                strcpy(str,&str[1]);
              else
                break;
            if (i <20)    /* f found */
            sscanf(&str[2],"%lf",&zforce);   /* get force from sensor */

        prints(0,str,50,1,0);
        sprintf(str,"%lf",zforce);
        prints(0,str,50,2,0);



        if (x < 12)                 /* check mins and maxs */
            x = 12;
          else if (x > 35.44)
            x = 35.44;

        if (z < -23)              /* do not allow crashing into floor */
            z = -23;

/* perform inverse kinematics for MERLIN */
```

fullsys.c

```
sinvkin(&wptr,x,y,z,wrist_roll,wrist_flex,tool_roll);
     }

     if (datcnt)
     {
          fp[0] = fopen("xvout.dat","w");
          fp[1] = fopen("yvout.dat","w");
          fp[2] = fopen("zvout.dat","w");
          for (i=0;i<3;i++)
               fprintf(fp[i],"%c%s\nTime\nvel in/sec\n133\n",'X'+i,"vout");

          for (i=0;i<datcnt-1;i++)
               for(j=0;j<3;j++)
                                                    /* calc velocity at 133 samples per sec */
                                   data[j][i] = (data[j][i] - data[j][i+1]) / 7.5187699e-3;

          for (i=0;i<datcnt-1;i++)
          {
               fprintf(fp[0],"%f\n",data[0][i]);
               fprintf(fp[1],"%f\n",data[1][i]);
               fprintf(fp[2],"%f\n",data[2][i]);
             }
             fclose(fp[0]);
             fclose(fp[1]);
             fclose(fp[2]);
        }

     close_status();
     tty_flush(1);
     tty_flush(2);
     tty_close(2);
     tty_close(1);
     if (done == 2)
          wind(8,8,"TIMEOUT",'E',' ',"Timeout waiting for MBA joint angles",
                                                            NULL);
                                   else if (done == 3)
          wind(8,8,"TIMEOUT",'E',' ',"Timeout waiting for JR3 data",
                                                            NULL);

}
```

transp.c

```
/************************************************************

  FILENAME:        transp.c

    FUNCTION NAME:    transpose.c

    AUTHOR:          Todd Mosher

  DATE:           4-26-91

    DESCRIPTION:

************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

int transpose(inmat,outmat)
double inmat[4][4];
double outmat[4][4];
{
        register int i,j;

    for (i=0;i<4;i++)
        for (j=0;j<4;j++)
            outmat[i][j] = inmat[j][i];
}
```

wristang.c

```
/***********************************************************

     FILENAME:       wristang.c

     FUNCTION NAME:   wrist_angles

     AUTHOR:         TW Mosher  and ML Crabill

   DATE:          4/26/91

     DESCRIPTION:

***********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "merlin.ref"


int wrist_angles()
{
        double mt0_5[4][4];
        double mata_5[4][4] = {0};
        double mat5_a[4][4] = {0};
        double mat9_a[4][4] = {0};
        char str[100];


            /*   first calculate the merlins 3 to 0 rotation matrix */
         merlrmat();

            /* calc a matrix to transform the MBA elbow to the merlin elbow */
         if (lock_wrist)
         {
          mt5_0[0][0] = -1;
          mt5_0[0][1] = 0;
          mt5_0[0][2] = 0;
          mt5_0[1][0] = 0;
          mt5_0[1][1] = 0;
          mt5_0[1][2] = -1;
          mt5_0[2][0] = 0;
          mt5_0[2][1] = -1;
          mt5_0[2][2] = 0;

/* for straight out  */
          mr9_5[0][0] = 0;
          mr9_5[0][1] = -1;
          mr9_5[0][2] = 0;
          mr9_5[1][0] = 0;
```

wristang.c

```
        mr9_5[1][1] = 0;
        mr9_5[1][2] = 1;
        mr9_5[2][0] = 0;
        mr9_5[2][1] = 0;
        mr9_5[2][2] = -1;

/* for straight down
        mr9_5[0][0] = -1;
        mr9_5[0][1] = 0;
        mr9_5[0][2] = 0;
        mr9_5[1][0] = 0;
        mr9_5[1][1] = 1;
        mr9_5[1][2] = 0;
        mr9_5[2][0] = 0;
        mr9_5[2][1] = 0;
        mr9_5[2][2] = -1;
*/

    }
    transpose(mt5_0,mt0_5);          /* transpose MBA elbow matrix */

    matmlt33(mt0_5,sr3_0,mata_5);  /* mult MERLIN elbow rotation */

/* matrix to the MBAs inverted */
/*matdisp(0,0,mata_5);               /* elbow rotation matrix */
    transpose(mata_5,mat5_a);


          /*  now calc the MBAs 9 to 5 matrix */

    matmlt33(mat5_a,mr9_5,mat9_a);
/*matdisp(0,5,mat9_a);*/


    wrist_flex = atan2(mat9_a[1][1],
                    -sqrt((double)(sqr(mat9_a[0][1]) + sqr(mat9_a[2][1]))));
          wrist_flex = atan2(-sqrt((double)(sqr(mat9_a[0][1]) + sqr(mat9_a[2][1]))),
                    mat9_a[1][1]);
          wrist_roll = atan2((double)(mat9_a[2][1] / sin(wrist_flex)),
                    (double)(-mat9_a[0][1] / sin(wrist_flex)));
          tool_roll = atan2((double)(mat9_a[1][2] / sin(wrist_flex)),
                   (double)(mat9_a[1][0] / sin(wrist_flex)));
/*    sprintf(str,"wr %7.2lf",(double)(wrist_roll*180.0/3.14159));
/*    prints(0,str,1,11,0);
/*    sprintf(str,"wf %7.2lf",(double)(wrist_flex *180.0/3.14159));
/*    prints(0,str,20,11,0);
/*    sprintf(str,"tr %7.2lf",(double)(tool_roll*180.0/3.14158));
/*    prints(0,str,40,11,0);
*/
```

wristang.c

```
    wrist_flex = atan2(mat9_a[1][1],
                       sqrt((double)(sqr(mat9_a[0][1]) + sqr(mat9_a[2][1]))));
            wrist_flex = atan2(sqrt((double)(sqr(mat9_a[0][1]) + sqr(mat9_a[2][1]))),
                       mat9_a[1][1]);
            wrist_roll = atan2((double)(mat9_a[2][1] / sin(wrist_flex)),
                       (double)(-mat9_a[0][1] / sin(wrist_flex)));

    tool_roll = atan2((double)(mat9_a[1][2] / sin(wrist_flex)),
                      (double)(mat9_a[1][0] / sin(wrist_flex)));
/*    sprintf(str,"wr %7.2lf",(double)(wrist_roll*180.0/3.14159));
/*    prints(0,str,1,10,0);
/*    sprintf(str,"wf %7.2lf",(double)(wrist_flex *180.0/3.14159));
/*    prints(0,str,20,10,0);
/*    sprintf(str,"tr %7.2lf",(double)(tool_roll*180.0/3.14158));
/*    prints(0,str,40,10,0);
*/

}
```

merlrmat.c

```
/************************************************************
          FILENAME:      merlrmat.c

          FUNCTION NAME:  merlrmat()

     AUTHOR:         Mosher & Crabill

      DATE:          04/26/91

************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

merlrmat()
{
double ct1,nct1,st1,nst1,ct2,nct2,st2,nst2,ct3,nct3,st3,nst3;


    ct1 = cos(theta[1]);
    nct1 = -ct1;
    st1 =  sin(theta[1]);
    nst1 = -st1;

    sr1_0[0][0] = ct1;
    sr1_0[0][1] = nst1;
    sr1_0[1][0] = nst1;
    sr1_0[1][1] = nct1;
    sr1_0[2][2] = -1;

/*  Merlin Left Arm Robot Matrix[2] to transform from coordinate system at Shoulder back to
       Reference. */

    ct2 = cos(theta[2]);
    nct2 = -ct2;
    st2 = sin(theta[2]);
    nst2 = -st2;

    sr2_0[0][0] = sr1_0[0][0] * ct2;
    sr2_0[0][1] = sr1_0[0][0] * nst2;
    sr2_0[0][2] = sr1_0[0][1];
    sr2_0[1][0] = sr1_0[1][0] * ct2;
    sr2_0[1][1] = sr1_0[1][0] * nst2;
    sr2_0[1][2] = sr1_0[1][1];
    sr2_0[2][0] = sr1_0[2][2] * nst2;
    sr2_0[2][1] = sr1_0[2][2] * nct2;
    sr2_0[2][2] = 0.0;
```

merlrmat.c

```
/*  Merlin Left Arm Robot Matrix[3] to transform from coordinate system at Elbow/Wrist Roll
       intersect back to Reference. */

    ct3 = cos(theta[3]);
    nct3 = -ct3;
    st3 = sin(theta[3]);
    nst3 = -st3;

    sr3_0[0][0] = sr2_0[0][0] * ct3 + sr2_0[0][1] * nst3;
    sr3_0[0][1] = sr2_0[0][0] * nst3 + sr2_0[0][1] * nct3;
    sr3_0[0][2] = -sr2_0[0][2];
    sr3_0[1][0] = sr2_0[1][0] * ct3 + sr2_0[1][1] * nst3;
    sr3_0[1][1] = sr2_0[1][0] * nst3 + sr2_0[1][1] * nct3;
    sr3_0[1][2] = -sr2_0[1][2];
    sr3_0[2][0] = sr2_0[2][0] * ct3 + sr2_0[2][1] * nst3;
    sr3_0[2][1] = sr2_0[2][0] * nst3 + sr2_0[2][1] * nct3;
    sr3_0[2][2] = 0;
}
```

jointang.c

```
/***********************************************************

    FILENAME:      jointang.c

    FUNCTION NAME:  joint_ang

    AUTHOR:        Todd Mosher

  DATE:          12-27-90

    MODIFIED:      23 Jan 91 by Montrose Crabbelly

    DESCRIPTION:
        This routine will get optical encoder information from
        the mba and then convert it to joint angles.

***********************************************************/
#include <stdio.h>
#include <conio.h>
#include "merlin.ref"

int joint_ang(wptr)
int wptr;
{
        int i;
        int lft_oe[8],rig_oe[8];
        char str[150];
        int itmp;
        double exo_hs_rad;

     get_mba_info(lft_oe,rig_oe,mba_rqst); /* get encoder information */
     for (i=0;i<7;i++)             /* calc angles of all joints */
          if (mba_rqst == 'L')
               exo_l_arm[i] = exo_l_hs_rad[i] + (lft_oe[i] - arm_hs_oe[i])
                                             * oe_to_rad[i];
          else
/*       exo_r_arm[i] = exo_r_hs_rad[i] + (arm_hs_oe[i] - rig_oe[i])
/*                  * oe_to_rad[i];
*/
               exo_r_arm[i] = exo_r_hs_rad[i] + (rig_oe[i] - arm_hs_oe[i])
                                             * oe_to_rad[i];

                                              /* dist from hard stop times the */
                                              /* conversion factor for encoder to deg */
                                              /* plus the hardstop position */

     prints(wptr,"Joint / Encoder / Angle / Encoder to rad / Hardstop(rad)"
                    " / Hardstop(cnts)",1,12,0);
for (i=0;i<7;i++)
    {
        if (mba_rqst == 'L')   /* get proper display information */
        {
            itmp = lft_oe[i];
            exo_hs_rad = exo_l_hs_rad[i];
          }
          else
          {
            itmp = rig_oe[i];
            exo_hs_rad = exo_r_hs_rad[i];
          }

    sprintf(str,"enc %2d  val %4d %7.2lf %7.3lf        %7.2lf
          %4d ", i,itmp,exo_r_arm[i] * 57.3, oe_to_rad[i],
          exo_hs_rad, arm_hs_oe[i]);
    prints(wptr,str,1,13+i,0);
  }

}
```

mbainit.c

```
/**************************************************************

    FILENAME:       mbainit.c

    FUNCTION NAME:  mba_init()

    AUTHOR:         Todd Mosher

  DATE:         12-31-90

    MODIFIED:       18 January 91 by Montrose Crabbelly

    DESCRIPTION:
        This routine will initialize any necessary variables.

**************************************************************/
#include <stdio.h>
#include <conio.h>
#include "merlin.ref"

int mba_init()
{
        FILE *fp;
        int i;

    if ((fp = fopen("mba.cfg","r")) == NULL)
    {
        tty_close(MBA_PORT);
        wn_exit();
        exit(wind(9,9,"FILE NOT FOUND",'E',' ',
                        "MBA configuration file missing", "Calibration should create this file",
                        "Press return!",
                        NULL));
            }
            else
        for (i=0;i<8;i++)
       fscanf(fp,"%d",&arm_hs_oe[i]);
       fclose(fp);
       mt2_0[0][2] = 0;
       mt2_0[0][3] = l3;
       mt2_0[1][2] = -.1564;
       mt2_0[1][3] = -(l1 + l4 * .1564);
       mt2_0[2][2] = -.9877;
       mt2_0[2][3] = -6.315 * (l1 + l4 * .1564)
                  + (l1 + l2 * .1584) / .1584;
                  x_offset = initx_offset;
                  y_offset = inity_offset;
                  z_offset = initz_offset;
}
```

rexotmat.c

```
/********************************************************
    FILENAME:     rexotmat.c

    FUNCTION NAME:  r_exo_tmat()

    AUTHOR:       Monty Crabill

  DATE:         06/18/92

    MODIFIED:

    DESCRIPTION:
                  Using MERLIN joint angle information, this creats the T matrix for the wrist
                  xyz position and the rotation matrix for the gripper with respect to the elbow.
                  The following method for calculating a final tmatrix is almost as fast as
                  crunching the matrixs by hand and coming up with
                  an equation (we know, we tested it).  However, this method is far easier to
                  modify, maintain, and read.

********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "merlin.ref"

/* double ct2,st2,nst2,ct3,nct3,st3,nst3,ct4,nct4,st4,nst4,
              ct5,nct5,st5,nst5,ct6,nct6,st6,nst6,ct7,st7,nst7,
              ct8,nct8,st8,nst8;
                                    /* nct7 is not used */
                                    /* used in tmat() to allow computation of */
                                    /* cos(*tX) & sin(*tX) only once */


r_exo_tmat()
{

/*  MBA Exo Right Matrix[2] to transform from coordinate system at */
/*  Right Shoulder/Azimuth Elevation Intersect back to Reference. */

    ct2 = cos(exo_r_arm[0]);
    st2 =  sin(exo_r_arm[0]);
    nst2 = -st2;

    mt2_0[0][0] = ct2;
    mt2_0[0][1] = nst2;
/* mt2_0[0][2] = 0;                  computed once in mbainit.c */
/* mt2_0[0][3] = 13;                 " */ mt2_0[1][0] = -.9877 * st2;
    mt2_0[1][1] = -.9877 * ct2;
```

rexotmat.c

```
/* mt2_0[1][2] = -.1564;                   " */
/* mt2_0[1][3] = -(l1 + l4 * .1564);           " */ mt2_0[2][0] = .1564 * st2;
     mt2_0[2][1] = .1564 * ct2;
/* mt2_0[2][2] = -.9877;                   " */
/* mt2_0[2][3] = -6.315 * (l1 + l4 * .1564)   "
                          + (l1 + l2 * .1584) / .1584; */

/*   MBA Exo Right Matrix[3] to transform from coordinate system at Right Shoulder Elevation/
     Upper Arm Roll Intersect back to Reference. */

     ct3 = cos(exo_r_arm[1]);
     nct3 = -ct3;
     st3 = sin(exo_r_arm[1]);
     nst3 = -st3;

     mt3_0[0][0] = mt2_0[0][0] * ct3 + mt2_0[0][2] * st3;
     mt3_0[0][1] = mt2_0[0][0] * nst3 + mt2_0[0][2] * ct3;
     mt3_0[0][2] = -mt2_0[0][1];
     mt3_0[0][3] = -mt2_0[0][1] * .032 + l3;
     mt3_0[1][0] = mt2_0[1][0] * ct3 + mt2_0[1][2] * st3;
     mt3_0[1][1] = mt2_0[1][0] * nst3 + mt2_0[1][2] * ct3;
     mt3_0[1][2] = -mt2_0[1][1];
     mt3_0[1][3] = -.032 * mt2_0[1][1] + mt2_0[1][3];
     mt3_0[2][0] = mt2_0[2][0] * ct3 + mt2_0[2][2] * st3;
     mt3_0[2][1] = mt2_0[2][0] * nst3 + mt2_0[2][2] * ct3;
     mt3_0[2][2] = -mt2_0[2][1];
     mt3_0[2][3] = -mt2_0[2][1] * .032 + mt2_0[2][3];

/*   MBA Exo Right Matrix[4] to transform from coordinate system at Right Upper Arm Roll/ Elbow
        Intersect back to Reference. */

     ct4 = cos(exo_r_arm[2]);
     nct4 = -ct4;
     st4 = sin(exo_r_arm[2]);
     nst4 = -st4;

     mt4_0[0][0] = mt3_0[0][0] * ct4 + mt3_0[0][2] * nst4;
     mt4_0[0][1] = mt3_0[0][0] * nst4 + mt3_0[0][2] * nct4;
     mt4_0[0][2] = mt3_0[0][1];
     mt4_0[0][3] = mt3_0[0][1] * l5 + mt3_0[0][3];
     mt4_0[1][0] = mt3_0[1][0] * ct4 + mt3_0[1][2] * nst4;
     mt4_0[1][1] = mt3_0[1][0] * nst4 + mt3_0[1][2] * nct4;
     mt4_0[1][2] = mt3_0[1][1];
     mt4_0[1][3] = mt3_0[1][1] * l5 + mt3_0[1][3];
     mt4_0[2][0] = mt3_0[2][0] * ct4 + mt3_0[2][2] * nst4;
     mt4_0[2][1] = mt3_0[2][0] * nst4 + mt3_0[2][2] * nct4;
```

```
    mt4_0[2][2] = mt3_0[2][1];
    mt4_0[2][3] = mt3_0[2][1] * l5 + mt3_0[2][3];

  /*  MBA Exo Right Matrix[5] to transform from coordinate system at Right Elbow and Lower
        Arm Roll Intersect back to Reference. */

    ct5 = cos(exo_r_arm[3]);
    nct5 = -ct5;
    st5 = sin(exo_r_arm[3]);
    nst5 = -st5;

    mt5_0[0][0] = mt4_0[0][0] * ct5 + mt4_0[0][2] * st5;
    mt5_0[0][1] = mt4_0[0][0] * nst5 + mt4_0[0][2] * ct5;
    mt5_0[0][2] = - mt4_0[0][1];
    mt5_0[0][3] = mt4_0[0][3];
    mt5_0[1][0] = mt4_0[1][0] * ct5 + mt4_0[1][2] * st5;
    mt5_0[1][1] = mt4_0[1][0] * nst5 + mt4_0[1][2] * ct5;
    mt5_0[1][2] = - mt4_0[1][1];
    mt5_0[1][3] = mt4_0[1][3];
    mt5_0[2][0] = mt4_0[2][0] * ct5 + mt4_0[2][2] * st5;
    mt5_0[2][1] = mt4_0[2][0] * nst5 + mt4_0[2][2] * ct5;
    mt5_0[2][2] = - mt4_0[2][1];
    mt5_0[2][3] = mt4_0[2][3];

/*  MBA Exo Right Matrix[6] to transform from coordinate system at Right
        Lower Arm Roll & Wrist Radial Intersect back to Reference. */

/*     ct6 = cos(exo_r_arm[4]);
/*     nct6 = -ct6;
/*     st6 = sin(exo_r_arm[4]);
/*     nst6 = -st6;


  /*   some element calcs have been eliminated since */
  /* only the position info from tmat 6 is used */

/*     mt6_0[0][0] = mt5_0[0][0] * ct6 + mt5_0[0][2] * st6;
/*     mt6_0[0][1] = mt5_0[0][0] * nst6 + mt5_0[0][2] * ct6;
/*     mt6_0[0][2] = -mt5_0[0][1];
*/
          mt6_0[0][3] = -mt5_0[0][1] * l6 + mt5_0[0][3];
/*     mt6_0[1][0] = mt5_0[1][0] * ct6 + mt5_0[1][2] * st6;
/*     mt6_0[1][1] = mt5_0[1][0] * nst6 + mt5_0[1][2] * ct6;
/*     mt6_0[1][2] = -mt5_0[1][1];
*/
          mt6_0[1][3] = -mt5_0[1][1] * l6 + mt5_0[1][3];
/*     mt6_0[2][0] = mt5_0[2][0] * ct6 + mt5_0[2][2] * st6;
/*     mt6_0[2][1] = mt5_0[2][0] * nst6 + mt5_0[2][2] * ct6;
/*     mt6_0[2][2] = -mt5_0[2][1];
*/
```

```
        mt6_0[2][3] = -mt5_0[2][1] * 16 + mt5_0[2][3];

/*  MBA Exo Right Matrix[7] to transform from coordinate system at Right
        Wrist Radial & Wrist Flex Intersect back to Reference. */
/* no longer needed */

/*      ct7 = cos(exo_r_arm[5]);
/*      nct7 = -ct7;
/*      st7 = sin(exo_r_arm[5]);
/*      nst7 = -st7;

/*      mt7_0[0][0] = mt6_0[0][0] * ct7 + mt6_0[0][2] * nst7;
/*      mt7_0[0][1] = mt6_0[0][0] * nst7 + mt6_0[0][2] * nct7;
/*      mt7_0[0][2] = mt6_0[0][1];
/*      mt7_0[0][3] = -mt6_0[0][1] * .160 + mt6_0[0][3];
/*      mt7_0[1][0] = mt6_0[1][0] * ct7 + mt6_0[1][2] * nst7;
/*      mt7_0[1][1] = mt6_0[1][0] * nst7 + mt6_0[1][2] * nct7;
/*      mt7_0[1][2] = mt6_0[1][1];
/*      mt7_0[1][3] = -mt6_0[1][1] * .160 +  mt6_0[1][3];
/*      mt7_0[2][0] = mt6_0[2][0] * ct7 + mt6_0[2][2] * nst7;
/*      mt7_0[2][1] = mt6_0[2][0] * nst7 + mt6_0[2][2] * nct7;
/*      mt7_0[2][2] = mt6_0[2][1];
/*      mt7_0[2][3] = -mt6_0[2][1] * .160 + mt6_0[2][3];
*/

/*  MBA Exo Right Matrix[8] to transform from coordinate system at Right
        Wrist Radial & Wrist Flex Intersect back to Reference. */
/*  no longer needed */
/*      ct8 = cos(exo_r_arm[6]);
/*      nct8 = -ct8;
/*      st8 = sin(exo_r_arm[6]);
/*      nst8 = -st8;

/*      mt8_0[0][0] = mt7_0[0][0] * ct8 + mt7_0[0][2] * nst8;
/*      mt8_0[0][1] = mt7_0[0][0] * nst8 + mt7_0[0][2] * nct8;
/*      mt8_0[0][2] = mt7_0[0][1];
/*      mt8_0[0][3] = mt7_0[0][3];
/*      mt8_0[1][0] = mt7_0[1][0] * ct8 + mt7_0[1][2] * nst8;
/*      mt8_0[1][1] = mt7_0[1][0] * nst8 + mt7_0[1][2] * nct8;
/*      mt8_0[1][2] = mt7_0[1][1];
/*      mt8_0[1][3] = mt7_0[1][3];
/*      mt8_0[2][0] = mt7_0[2][0] * ct8 + mt7_0[2][2] * nst8;
/*      mt8_0[2][1] = mt7_0[2][0] * nst8 + mt7_0[2][2] * nct8;
/*      mt8_0[2][2] = mt7_0[2][1];
/*      mt8_0[2][3] = mt7_0[2][3];
*/
```

rexotmat.c

```
/****************************
*     rotation matrix from the wrist to the elbow
*
****************************/

    ct6 = cos(exo_r_arm[4]);
    st6 = sin(exo_r_arm[4]);

/*       matrix to describe wrist roll to the elbow */

    mr6_5[0][0] = ct6;
    mr6_5[0][1] = -st6;
    mr6_5[1][2] = 1;
    mr6_5[2][0] = -st6;
    mr6_5[2][1] = -ct6;

/*       matrix to describe wrist flex to the elbow */

    ct7 = cos(exo_r_arm[5]);
    st7 = sin(exo_r_arm[5]);

    mr7_5[0][0] = mr6_5[0][0] * ct7;
    mr7_5[0][1] = mr6_5[0][0] * -st7;
    mr7_5[0][2] = mr6_5[0][1];
    mr7_5[1][0] = st7;
    mr7_5[1][1] = ct7;
    mr7_5[1][2] = 0;
    mr7_5[2][0] = mr6_5[2][0] * ct7;
    mr7_5[2][1] = mr6_5[2][0] * -st7;
    mr7_5[2][2] = mr6_5[2][1];


/*   matrix to describe tool roll to the elbow */

    ct8 = cos(exo_r_arm[6]);
    st8 = sin(exo_r_arm[6]);

    mr8_5[0][0] = mr7_5[0][0] * ct8 + mr7_5[0][2] * -st8;
    mr8_5[0][1] = mr7_5[0][0] * -st8 + mr7_5[0][2] * -ct8;
    mr8_5[0][2] = mr7_5[0][1];
    mr8_5[1][0] = mr7_5[1][0] * ct8;
    mr8_5[1][1] = mr7_5[1][0] * -st8;
    mr8_5[1][2] = mr7_5[1][1];
    mr8_5[2][0] = mr7_5[2][0] * ct8 + mr7_5[2][2] * -st8;
    mr8_5[2][1] = mr7_5[2][0] * -st8 + mr7_5[2][2] * -ct8;
    mr8_5[2][2] = mr7_5[2][1];

/*   matrix to orient the frame 8 the same as the merlin gripper   */


    mr9_5[0][0] = mr8_5[0][2];
    mr9_5[0][1] = -mr8_5[0][0];
    mr9_5[0][2] = -mr8_5[0][1];
    mr9_5[1][0] = mr8_5[1][2];
    mr9_5[1][1] = -mr8_5[1][0];
    mr9_5[1][2] = -mr8_5[1][1];
    mr9_5[2][0] = mr8_5[2][2];
    mr9_5[2][1] = -mr8_5[2][0];
    mr9_5[2][2] = -mr8_5[2][1];

}
```

calib.c

```
/*********************************************************

     FILENAME:        calib.c

     FUNCTION NAME:   calib_mba()

     AUTHOR:          Todd Mosher

   DATE:          12-31-90

     DESCRIPTION:
        This routine will instuct the user how to calibrate the
        mba.

*********************************************************/
#include <stdio.h>
#include <conio.h>
#include "merlin.ref"

extern int tty_cnt;
char cal_txt[8][25] =
               { {"shoulder azimuth"},
                 {"shoulder elevation"},
                 {"upper arm roll"},
                 {"elbow flex"},
                 {"lower arm roll"},
                 {"wrist roll"},
                 {"wrist flex"},
                 {"          "}
                };



int calib_mba()
{
        FILE *fp;
        int i,j;
        char str[100];
        char str2[100];
        int lft[8],right[8];
        int res[8];

     wind(8,2,"INFORMATION",'I',' ',
             "All rotations are clock wise. Look down the ",
             "optical encoder shaft into the housing to ",
             "determine the proper direction",NULL);


     for (i=0;i<7;i++)    /* for each joint */
     {
         sprintf(str,"Please rotate the %s",cal_txt[i]);
```

calib.c

```
        if (mba_rqst == 'R')            /* display proper arm request */ sprintf(str2,"to the hardstop
            (right arm).");
          else
            sprintf(str2,"to the hardstop (left arm).");

        wind(10,8,"INSTRUCTIONS",'E',' ',
        str,
        str2,
        "Press enter when this is done.",
        NULL);
        get_mba_info(lft,right,mba_rqst); /* this gets data from old data request */
        get_mba_info(lft,right,mba_rqst);   /* this gets current data */

        if (mba_rqst == 'L')          /* save proper results  */
            res[i] = lft[i];          /* save result in res */
          else
            res[i] = right[i];          /* save result in res */
        }
        close_info();

    fp = fopen("mba.cfg","w");   /* save the current info */
    for (i=0;i<8;i++)
        fprintf(fp,"%d\n",res[i]);
      fclose(fp);
}
```

getmba.c

```
/************************************************************

  FILENAME:       getmba.c

    FUNCTION NAME:    get_mba_info()

    AUTHOR:         Todd Mosher

  DATE:          12-31-90

    DESCRIPTION:
        This routine will get the optical encoder information
        from the mba

************************************************************/
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include "merlin.ref"

extern int trigr,datcnt;

int get_mba_info(lft_oe,rig_oe,typ)
int *lft_oe,*rig_oe;
int typ;              /* Left Right or H/Both */
{
        char tbuff[32];    /* temp input buffer */
        char byt;
        int j,i;
        int bytecnt = 17;
        char str[100];
        unsigned int tm;

    typ = toupper(typ);
    if (typ == 'B' || typ == 'H')
        bytecnt = 33;

    tm = utime();
    while(tty_cnt(MBA_PORT) < bytecnt) /* wait for all data to return from mba */
        if (tm - utime() >= 25000)      /* timeout so getout */
        {
            if (kbhit() && getch() == 27)
            {
                done = 2;
                return;
              }
              tty_open(MBA_PORT,10,8,1,0);        /* setup the serial port for mba */
              sprintf(str,"%d Missing MBA data",++badcnt);
              prints(0,str,25,1,0);
```

getmba.c

```
    while (tty_cnt(MBA_PORT))  /* clear the port */
                tty_in(MBA_PORT);
              tty_out(MBA_PORT,typ);    /* send out new request for data */
              tm = utime();       /* setup time for new request */
         }
      indexing = 0;                          /* turn off indexing */
      byt = tty_in(MBA_PORT);                       /* get leading byte */
      if (byt == 'R')                        /* index reset */
      {
        x_offset = initx_offset;
        y_offset = inity_offset;
        z_offset = initz_offset;
      }
      else if (byt == 'I')                   /* turn on indexing */
        indexing = 1;
      else if (byt != 'H' && byt != 'Z')   /* h,i,r,z are valid header bytes */
      {                             /* z is halt button */
        while(tty_cnt(MBA_PORT))       /* clear the port */
            tty_in(MBA_PORT);
          sprintf(str,"%d Bad header found",++badcnt);
          prints(0,str,25,1,0);
      }
      if (typ == 'H' || typ == 'B')
        for (j=31;j>=0;j--)                /* get the data */
            tbuff[j] = tty_in(MBA_PORT);
        else if (typ == 'L')
        for (j=31;j>=16;j--)
            tbuff[j] = tty_in(MBA_PORT);
        else if (typ == 'R')
        for (j=15;j>=0;j--)
            tbuff[j] = tty_in(MBA_PORT);
        tty_out(MBA_PORT,typ);              /* give next request to mba */
        if (byt == 'Z')         /* when in halt, do not update numbers */ return;
        if (typ == 'B' || typ == 'H' || typ == 'R')
        for (j=7;j>=0;j--)    /* put highbyte,lowbyte together */
            rig_oe[j]= ((tbuff[2*j+1]<<8) & 0xFF00) | (tbuff[2*j] & 0x00FF);
        if (typ == 'B' || typ == 'H' || typ == 'L')
        for (j=15;j>=8;j--)
            lft_oe[j-8]=((tbuff[2*j+1]<<8) & 0xFF00) | (tbuff[2*j] & 0x00FF);
/*   gotoxy(1,3);
     printf("%5d  %5d  %5d %5d\n",rig_oe[0],rig_oe[1],rig_oe[2],rig_oe[3]); printf("%5d  %5d
     %5d  %5d\n",rig_oe[4],rig_oe[5],rig_oe[6],rig_oe[7]);
*/
}
```

ttyopen.c

```
/*
TTYOPEN.C

Task 25, Robotics

KL Johnston, SRL

Functions used by TTY68K to initialize and restore IBM-PC serial ports.
A serial input pattern matching function is also available.
*/

#include <bios.h>
#include <dos.h>
#include <stdio.h>

#define _1STOP      0x00
#define _2STOP      0x04
#define _7BITS      0x02
#define _8BITS      0x03
#define BASE1       0x3F8
#define BASE2       0x2E8
#define DLAB        0x80
#define DTR         0x01
#define IRQ1        0x10
#define IRQ2        0x20   /* 0x08  */
#define OUT1        0x04
#define OUT2        0x08
#define RTS         0x02
#define VECTOR1     0x0C
#define VECTOR2     0x0D   /* 0x0B  */
int port = 1;           /* com port ... default com1 */
int baud = 8;           /* index into baud table... default 9600 */
int dbits = 8;          /* data bits... default 8 */
int sbits = 1;          /* stop bits... default 1 */
int parity = 0;  /* parity.. default none */
int baud_table[12][3] =         /* baud table for serial io stuff */
{
    {   50, 9,   0},
    {  110, 4,  23},
    {  150, 3,   0},
    {  300, 1, 128},
    {  600, 0, 192},
    { 1200, 0,  96},
    { 2400, 0,  48},
    { 4800, 0,  24},
    { 9600, 0,  12},
    {19200, 0,   6},
    {38400, 0,   3},
    {56000, 0,   2}
};
```

ttyopen.c

```
void interrupt tty1_isr(void);
void interrupt tty2_isr(void);
extern int tty1_base, tty1_error, tty1_qfull,tty1_cnt;
extern int tty2_base, tty2_error, tty2_qfull,tty2_cnt;


static int irq_level, vector;
static void interrupt (*old_isr)();

/************************************************************************ ******/
/********************** t t y _ o p e n ***********************************/
/************************************************************************ ******/ void
tty_open(int port,int baud,int dbits,int sbits,int parity)
/*
   Initializes serial port and sets tty_isr (see TTYISR.ASM) as interrupt
   service routine.
   port   = 1 - 2 for COM1 or COM2 port.
   baud   = 0 - 11, index into baud_table array.
   dbits  = 7 - 8, number of data bits/byte.
   sbits  = 1 - 2, number of stop bits/byte.
   parity = EVEN_PARITY, ODD_PARITY or NO_PARITY (see TTYOPEN.H).
   This routine is not intended to be fool proof, it assumes valid arguments
   and returns no status.
*/
{
        int w_len, n_stop;
        int tty_base;


        if (port == 1)
        {
                tty1_base  = BASE1;
                tty_base   = BASE1;
                vector     = VECTOR1;
                irq_level = IRQ1;
            }
            else
            {
                tty2_base  = BASE2;
                tty_base   = BASE2;
                vector     = VECTOR2;
                irq_level = IRQ2;
            }
            if (dbits == 7)
                w_len = _7BITS;
            else
                w_len = _8BITS;
            if (sbits == 1)
            n_stop = _1STOP;
        else
            n_stop = _2STOP;
        old_isr = getvect(vector);
        if (port == 1)
         setvect(vector,tty1_isr); /* Establish 8259 handler for tty IRQ level */ else
         setvect(vector,tty2_isr); /* Establish 8259 handler for tty IRQ level */

    disable();
    outportb(tty_base+4,RTS | DTR);  /* Disable any pending interrupts */
    outportb(tty_base+1,0);
    outportb(tty_base+3,DLAB | w_len | n_stop | parity);
    outportb(tty_base,baud_table[baud][2]); outportb(tty_base+1,baud_table[baud][1]);
    outportb(tty_base+3,w_len | n_stop | parity);
    inportb(tty_base);          /* Clear any pending 8250 receive */
    inportb(tty_base+5);        /*                    or status interrupts */
    inportb(tty_base+6);        /*                    or modem interrupts  */
    outportb(tty_base+1,0x07); /* Toggle enable bits to activate */ outportb(tty_base+1,0);
    outportb(0x21,~irq_level & inportb(0x21)); /* Enable 8259 interrupts */
    outportb(tty_base+4,OUT2 | RTS | DTR); /* OUT2 is wired as int enable */
    outportb(tty_base+1,0x07); /* Enable recv, xmit & status interrupts */ enable();
}

/*************************************************************************** ******/
/************************* t t y   c l o s e *******************************/
/*************************************************************************** ******/ void
tty_close(port)
int port;
/*
   Disables all interrupts from serial port previously initialized (tty_open).  Restores original interrupt
   service routine for serial port.
   tty_open must have been previously called.
   If program exits without calling this routine, the computer will likely
   crash.
*/
{
   int tty_base;

      if (port == 1)
          tty_base = tty1_base;
        else
```

```
            tty_base = tty2_base;

        disable();
        outportb(0x21,irq_level | inportb(0x21));
        outportb(tty_base+4,RTS | DTR);
        outportb(tty_base+1,0);
        enable();
        tty_flush(port);
        setvect(vector,old_isr);
}
/*************************
*
*     Support routines for selective port handling
*
**************************/
int tty_cnt(port)
int port;
{
    if (port == 1)          /* return char from specified port */ return(tty1_cnt);
    return(tty2_cnt);
}
int tty_qfull(port)
int port;
{
    if (port == 1)          /* return char from specified port */ return(tty1_qfull);
    return(tty2_qfull);
}
int tty_error(port)
int port;
{
    if (port == 1)          /* return char from specified port */ return(tty1_error);
    return(tty2_error);
}
int tty_in(port)
int port;
{
    if (port == 1)          /* return char from specified port */ return(tty1_in());
    return(tty2_in());
}
int tty_out(port,byt)
int port,byt;
{
    if (port == 1)
        return(tty1_out(byt));
      return(tty2_out(byt));
}
int tty_outs(port,str)
```

ttyopen.c

```
int port;
char *str;
{
    if (port == 1)
        return(tty1_outs(str));
      return(tty2_outs(str));
}
int tty_outmem(port,str,cnt)
int port;
int *str;
int cnt;
{
    if (port == 1)
        return(tty1_outmem(str,cnt));
      return(tty2_outmem(str,cnt));
}
int tty_flush(port)
int port;
{
    if (port == 1)
        return(tty1_flush());
      return(tty2_flush());
}


/***************************************************************** ******/
/*********************** t t y _ i n _ m a t c h **************************/
/***************************************************************** ******/ int
tty_in_match(char *pattern,int timeout_seconds)
/*
  Tries to match incoming tty character stream to a pattern string.
  When a mismatch is found the matching process is restarted.
  When a complete match is found, return value is true.
  If timeout expires before a match is found, return value is false.
*/
{
      int ch, i;
      long timeout;

      timeout = biostime(0,0) + 18 * timeout_seconds;
      i = 0;
      while (pattern[i] != 0)
      {
            if (tty_error || tty_qfull || biostime(0,0) > timeout)
                  return(0);
               if ((ch = tty_in()) != EOF && ch != (pattern[i++] & 0xFF))
                  i = 0;
            }
            return(1);
}
```

ttylisr.asm

```
                name    ttyisr
;             large memory model
;
;          **********   this version has no Xon Xoff stuff   todd
mosher
; TTYISR.ASM
;
; Task 25, Robotics
;
; KL Johnston, SRL
;
; Turbo C functions used by TTY68K. Interrupt service and input/output
; queue interface routines for serial (COM) port communications.
; Used along with functions in TTYOPEN.C.
;

EOF        = -1
EOI        = 20h
INQ_SIZE  = 200h       ; was  256
NO_SERV    = 01h
OUTQ_SIZE = 2020h   ;Que size limits max array tty_outs & tty_outmem can handle.
RECV_SERV = 04h
XMIT_SERV = 02h
XOFF       = 13h
XON        = 11h

                public  _tty1_base
                public  _tty1_error
                public  _tty1_flush
                public  _tty1_in
                public  _tty1_isr
                public  _tty1_out
                public  _tty1_outmem
                public  _tty1_outs
                public  _tty1_qfull
                public  _tty1_cnt

                public  _tty2_base
                public  _tty2_error
                public  _tty2_flush
                public  _tty2_in
                public  _tty2_isr
                public  _tty2_out
                public  _tty2_outmem
                public  _tty2_outs
                public  _tty2_qfull
                public  _tty2_cnt

_data   segment word public 'data'
in1_head          dw 0
in1_tail         dw 0
```

ttylisr.asm

```
out1_head       dw 0
out1_tail      dw 0
xmit1_idle      dw 0
send1_xoff      dw 0
send1_xon       dw 0
recv1_pause     dw 0
_tty1_base      dw 0
_tty1_error     dw 0
_tty1_qfull     dw 0
_tty1_cnt       dw 0

in2_head        dw 0
in2_tail       dw 0
out2_head       dw 0
out2_tail      dw 0
xmit2_idle      dw 0
send2_xoff      dw 0
send2_xon       dw 0
recv2_pause     dw 0
_tty2_base      dw 0
_tty2_error     dw 0
_tty2_qfull     dw 0
_tty2_cnt       dw 0
_data   ends

_bss    segment word public 'bss'
inq1     db INQ_SIZE  dup (?)
outq1    db OUTQ_SIZE dup (?)
inq2     db INQ_SIZE  dup (?)
outq2    db OUTQ_SIZE dup (?)
_bss    ends

_text   segment byte public 'code'
dgroup  group   _data,_bss
                assume  cs:_text,ds:dgroup,ss:dgroup

_tty1_isr       proc    near
;
; Serial port interrupt service routine. Never directly called from program.
;
        push    ax              ;Save a few registers to work with.
        push    bx
        push    dx
        push    ds
        mov     ax,dgroup       ;Set up ds to access Turbo C objects.
        mov     ds,ax
$20:
        mov     dx,_tty1_base   ;Read interrupt identification
register.
        add     dx,2
```

ttylisr.asm

```
          in      al,dx
          cmp     al,NO_SERV       ;Any service required?
          jne     $30              ;Yes.
          jmp     done             ;No.
$30:
        cmp     al,RECV_SERV     ;Service received data interrupt?
        jne     not1_recv        ;No.
        mov     dx,_tty1_base    ;Yes, read character.
        in      al,dx
;        or      al,al            ;Null character?
;        jz      $20              ;Yes, ignore it.
;        cmp     al,XOFF          ;No, Xoff character?
;        je      $20              ;Yes, ignore it.
;        cmp     al,XON           ;No, Xon character?
;        je      $20              ;Yes, ignore it.
        mov     bx,_tty1_cnt     ; increment the
        inc     bx               ; byte count
        mov     _tty1_cnt,bx     ; and save it
        mov     bx,in1_head      ;Advance input que head pointer.
        inc     bx
        cmp     bx,INQ_SIZE
        jl      $100
        xor     bx,bx
$100:
        cmp     bx,in1_tail      ;Input que full?
        je      $200             ;Yes.
        mov     in1_head,bx      ;No, save new que head pointer.
        mov     inq1[bx],al      ;Put received character on que.
        cmp     recv1_pause,0    ;Receiver pause already flagged?
        jne     $20              ;Yes, don't bother checking again.
        mov     ax,in1_tail      ;Calculate free space remaining.
        sub     ax,bx
        dec     ax
        jge     $140
        add     ax,INQ_SIZE
$140:
;        cmp     ax,INQ_SIZE/4    ;Input que 75% full?
;        jge     $20              ;No.
;        mov     recv1_pause,1    ;Yes, initiate receiver pause.
;        mov     send1_xoff,1
;        call    tickle1_xmit
        jmp     $20
$200:
        inc     _tty1_qfull      ;Count que full error.
        jmp     $20              ;Check for more servicing.
not1_recv:
        cmp     al,XMIT_SERV     ;Service transmit ready interrupt?
        jne     not1_xmit        ;No.
        cmp     send1_xoff,0     ;Yes, send Xoff?
        je      $250             ;No.
        mov     send1_xoff,0     ;Yes, reset flag.
```

```
          mov    al,XOFF
          jmp    $420
$250:
      cmp    send1_xon,0     ;Send Xon?
      je     $260            ;No.
      mov    send1_xon,0     ;Yes, reset flag.
      mov    al,XON
      jmp    $420
$260:
      cmp    recv1_pause,0   ;Receiver ISR paused?
      jne    $290            ;Yes, transmission must also pause.
      mov    bx,out1_tail    ;No, output que empty?
      cmp    bx,out1_head
      jne    $300            ;No.
$290:
      mov    xmit1_idle,1    ;Yes, flag outport idle.
      jmp    $20             ;Check for more servicing.
$300:
      inc    bx              ;Advance que tail pointer.
      cmp    bx,OUTQ_SIZE
      jl     $400
      xor    bx,bx
$400:
      mov    out1_tail,bx    ;Save new que tail pointer.
      mov    al,outq1[bx]    ;Get character to be transmitted.
$420:
      mov    dx,_tty1_base
      out    dx,al           ;Transmit character.
      jmp    $20             ;Check for more servicing.
not1_xmit:
      mov    dx,_tty1_base   ;Assume status interrupt.
      add    dx,5
      in     al,dx           ;Access line status reg to clear
      error.
      inc    dx
      in     al,dx           ;Access modem status reg to clear
      error.
      inc    _tty1_error     ;Count hardware error.
      jmp    $20             ;Check for more servicing.
done:
      mov    al,EOI          ;Non-specific EOI for 8259.
      out    20h,al
      pop    ds              ;Restore registers.
      pop    dx
      pop    bx
      pop    ax
      iret
_tty1_isr      endp

_tty1_in proc    far
;
```

ttylisr.asm

```
; Returns next character on input que to caller as an integer.
; If input que is empty, returns EOF (-1).
;
        mov     ax,EOF          ;Assume que empty.
        mov     bx,in1_tail
        cmp     bx,in1_head     ;Input que empty?
        je      $1200           ;Yes.
        cli
        mov     bx,_tty1_cnt    ; decrement the
        dec     bx              ; byte count
        mov     _tty1_cnt,bx    ; and save it
        sti
        mov     bx,in1_tail
        inc     bx              ;No, advance input que tail pointer.
        cmp     bx,INQ_SIZE
        jl      $1100
        xor     bx,bx
$1100:
        mov     al,inq1[bx]     ;Return next character from input
que.
        xor     ah,ah
        mov     in1_tail,bx     ;Save new tail pointer.
$1200:
        push    ax              ;Save return value.
        cmp     recv1_pause,0   ;Receiver paused?
        je      $1900           ;No.
        mov     ax,in1_head     ;Yes, calculate characters left in
        que.
        sub     ax,bx
        jge     $1300
        add     ax,INQ_SIZE
$1300:
        cmp     ax,INQ_SIZE/4   ;Input que 75% free?
        jge     $1900           ;No.
        cli                     ;Yes, no interrupts while changing
        pause.
        mov     recv1_pause,0   ;Clear receiver ISR pause.
        mov     send1_xon,1
        call    tickle1_xmit
        sti
$1900:
        pop     ax
        ret
_tty1_in        endp
_tty1_out       proc    far
;
; Adds a character (supplied by caller as an integer) to output que.
; If output que is full, character is not queued and function
; returns false (0), else function returns true (1).
;
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
;       cmp     byte ptr [bp+6],XOFF ;Caller sending Xoff?
;       je      $2800           ;Yes, ignore it.
        mov     bx,out1_head    ;No, advance output que head pointer.
        inc     bx
        cmp     bx,OUTQ_SIZE
        jl      $2300
        xor     bx,bx
$2300:
        xor     ax,ax           ;Assume failure.
        cmp     bx,out1_tail    ;Output que full?
        je      $2900           ;Yes, return failure.
        mov     al,[bp+6]       ;No, get character passed by caller.
        mov     outq1[bx],al    ;Add character to output que.
        mov     out1_head,bx    ;Save new head pointer.
        cli
        call    tickle1_xmit
        sti
$2800:
        mov     ax,1            ;Return success.
$2900:
        pop     bp
        ret
_tty1_out       endp

_tty1_outs      proc    far
;
; Adds an entire null terminated string (supplied by caller) to output que.
; If string is longer than output que, false (0) is returned.
; If there is currently insufficient free space for the entire string to
; be placed on the output que, false (0) is returned.
; If string length is zero, true (1) is returned, but no action takes place.
; Else string is placed on output que and true (1) is returned.
;
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
        push    si              ;Save callers index registers.
        push    di
        push    ds
        pop     es              ;Pointer ES:DI to callers string.
        mov     di,[bp+6]
        mov     cx,OUTQ_SIZE+2  ;Max legal string length including
        null + 1.
        xor     ax,ax           ;0 = end of string.
        cld                     ;Make sure we increment DI.
        repnz scasb             ;String too long?
        jcxz    $3800           ;Yes.
        sub     cx,OUTQ_SIZE+1  ;No, calculate length of string.
```

```
            neg     cx
            je      $3700           ;Zero length string, all done.
$3000:
        mov     ax,out1_tail    ;Calculate free space assuming tail >
head.
        sub     ax,out1_head
        dec     ax
        jge     $3100
        add     ax,OUTQ_SIZE    ;Tail < = head, adjust for que wrap
around. $3100:
        cmp     ax,cx           ;Enough free space for entire string?
        jl      $3800           ;No.
        mov     si,[bp+6]       ;Yes, set up to xfer string to que.
        lea     di,outq1
        add     di,out1_head
        inc     di
$3200:
        cmp     di,offset dgroup:outq1+OUTQ_SIZE ;Need to wrap pointer
around?
        jl      $3300           ;No.
        lea     di,outq1        ;Yes.
$3300:
        movsb                   ;Transfer one byte to output que.
        loop    $3200
        dec     di
        sub     di,offset dgroup:outq1 ;Calculate new que head
        pointer.
        mov     out1_head,di    ;Save new que head.
        cli
        call    tickle1_xmit
        sti
$3700:
        mov     ax,1            ;Return success.
        jmp     $3900
$3800:
        xor     ax,ax           ;Return failure.
$3900:
        pop     di
        pop     si
        pop     bp
        ret
;
; Alternate entry for _tty_outs. Queues an array of bytes the size of
which
; is specified by the caller.
;
_tty1_outmem:
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
        push    si              ;Save callers index registers.
        push    di
```

ttylisr.asm

```
            push   ds
            pop    es              ;Initialize ES = DS.
            mov    cx,[bp+6]       ;Get callers byte count.
            or     cx,cx           ;Byte count less than or equal zero?
            jl     $3800           ;Yes, illegal.
            je     $3700           ;Yes, all done.
            cmp    cx,OUTQ_SIZE    ;No, byte count too high?
            jg     $3800           ;Yes.
            jmp    $3000           ;No.
_tty1_outs     endp

_tty1_flush    proc    far
;
; Reset all input and output queue pointers (losing any characters
; currently in either queue). Also resets error counters.
; No arguments or return values.
;
            xor    ax,ax
            cli
            mov    in1_head,ax       ;Reset input que.
            mov    in1_tail,ax
            mov    out1_head,ax      ;Reset output que.
            mov    out1_tail,ax
            mov    _tty1_cnt,ax      ;reset byte count
            mov    xmit1_idle,1
            mov    _tty1_error,ax    ;Reset error counts.
            mov    _tty1_qfull,ax
            mov    send1_xoff,ax     ;Reset Xon/Xoff flags.
            mov    send1_xon,ax
            mov    recv1_pause,ax
            sti
            ret
_tty1_flush    endp

tickle1_xmit   proc   near
       cmp    xmit1_idle,0     ;Transmitter ISR idle?
       je     $5800            ;No.
       mov    dx,_tty1_base    ;Yes, toggle xmit interrupt enable to
       inc    dx               ;force xmit interrupt.
       in     al,dx
       mov    bl,al
       and    al,0FDh
       out    dx,al
       mov    al,bl
       out    dx,al
       mov    xmit1_idle,0
$5800:
       ret
tickle1_xmit   endp
```

ttylisr.asm

```
;***************** routines for port 2 ***************
_tty2_isr       proc    near
;
; Serial port interrupt service routine. Never directly called from program.
;
        push    ax              ;Save a few registers to work with.
        push    bx
        push    dx
        push    ds
        mov     ax,dgroup       ;Set up ds to access Turbo C objects.
        mov     ds,ax
$22:
        mov     dx,_tty2_base   ;Read interrupt identification
register.
        add     dx,2
        in      al,dx
        cmp     al,NO_SERV      ;Any service required?
        jne     $32             ;Yes.
        jmp     done2           ;No.
$32:
        cmp     al,RECV_SERV    ;Service received data interrupt?
        jne     not2_recv       ;No.
        mov     dx,_tty2_base   ;Yes, read character.
        in      al,dx
;       or      al,al           ;Null character?
;       jz      $22             ;Yes, ignore it.
;       cmp     al,XOFF         ;No, Xoff character?
;       je      $22             ;Yes, ignore it.
;       cmp     al,XON          ;No, Xon character?
;       je      $22             ;Yes, ignore it.
        mov     bx,_tty2_cnt    ; increment the
        inc     bx              ; byte count
        mov     _tty2_cnt,bx    ; and save it
        mov     bx,in2_head     ;Advance input que head pointer.
        inc     bx
        cmp     bx,INQ_SIZE
        jl      $102
        xor     bx,bx
$102:
        cmp     bx,in2_tail     ;Input que full?
        je      $202            ;Yes.
        mov     in2_head,bx     ;No, save new que head pointer.
        mov     inq2[bx],al     ;Put received character on que.
        cmp     recv2_pause,0   ;Receiver pause already flagged?
        jne     $22             ;Yes, don't bother checking again.
        mov     ax,in2_tail     ;Calculate free space remaining.
        sub     ax,bx
        dec     ax
        jge     $142
        add     ax,INQ_SIZE
$142:
```

```
;        cmp     ax,INQ_SIZE/4   ;Input que 75% full?
;        jge     $22             ;No.
;        mov     recv2_pause,1   ;Yes, initiate receiver pause.
;        mov     send2_xoff,1
;        call    tickle2_xmit
         jmp     $22
$202:
         inc     tty2_qfull      ;Count que full error.
         jmp     $22             ;Check for more servicing.
not2_recv:
         cmp     al,XMIT_SERV    ;Service transmit ready interrupt?
         jne     not2_xmit       ;No.
         cmp     send2_xoff,0    ;Yes, send Xoff?
         je      $252            ;No.
         mov     send2_xoff,0    ;Yes, reset flag.
         mov     al,XOFF
         jmp     $422
$252:
         cmp     send2_xon,0     ;Send Xon?
         je      $262            ;No.
         mov     send2_xon,0     ;Yes, reset flag.
         mov     al,XON
         jmp     $422
$262:
         cmp     recv2_pause,0   ;Receiver ISR paused?
         jne     $292            ;Yes, transmission must also pause.
         mov     bx,out2_tail    ;No, output que empty?
         cmp     bx,out2_head
         jne     $302            ;No.
$292:
         mov     xmit2_idle,1    ;Yes, flag outport idle.
         jmp     $22             ;Check for more servicing.
$302:
         inc     bx              ;Advance que tail pointer.
         cmp     bx,OUTQ_SIZE
         jl      $402
         xor     bx,bx
$402:
         mov     out2_tail,bx    ;Save new que tail pointer.
         mov     al,outq2[bx]    ;Get character to be transmitted.
$422:
         mov     dx, tty2_base
         out     dx,al           ;Transmit character.
         jmp     $22             ;Check for more servicing.
not2_xmit:
         mov     dx, tty2_base   ;Assume status interrupt.
         add     dx,5
         in      al,dx           ;Access line status reg to clear
         error.
         inc     dx
         in      al,dx   ;Access modem status reg to clear error.
```

```
inc     _tty2_error     ;Count hardware error.
        jmp    $22             ;Check for more servicing.
done2:
        mov    al,EOI          ;Non-specific EOI for 8259.
        out    20h,al
        pop    ds              ;Restore registers.
        pop    dx
        pop    bx
        pop    ax
        iret
_tty2_isr       endp

_tty2_in proc   far
;
; Returns next character on input que to caller as an integer.
; If input que is empty, returns EOF (-1).
;
        mov     ax,EOF          ;Assume que empty.
        mov     bx,in2_tail
        cmp     bx,in2_head     ;Input que empty?
        je     $1202            ;Yes.
        cli
        mov     bx,_tty2_cnt    ; decrement the
        dec    bx               ; byte count
        mov     _tty2_cnt,bx    ; and save it
        sti
        mov     bx,in2_tail
        inc    bx               ;No, advance input que tail pointer.
        cmp     bx,INQ_SIZE
        jl     $1102
        xor    bx,bx
$1102:
        mov     al,inq2[bx]     ;Return next character from input
que.
        xor    ah,ah
        mov     in2_tail,bx     ;Save new tail pointer.
$1202:
        push   ax               ;Save return value.
        cmp     recv2_pause,0   ;Receiver paused?
        je     $1902            ;No.
        mov     ax,in2_head     ;Yes, calculate characters left in
        que.
        sub    ax,bx
        jge    $1302
        add    ax,INQ_SIZE
$1302:
        cmp     ax,INQ_SIZE/4   ;Input que 75% free?
        jge    $1902            ;No.
        cli                     ;Yes, no interrupts while changing
        pause.
        mov     recv2_pause,0   ;Clear receiver ISR pause.
```

```
            mov     send2_xon,1
            call    tickle2_xmit
            sti
$1902:
            pop     ax
            ret
_tty2_in        endp
_tty2_out       proc    far
;
; Adds a character (supplied by caller as an integer) to output que.
; If output que is full, character is not queued and function
; returns false (0), else function returns true (1).
;
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
;        cmp    byte ptr [bp+6],XOFF ;Caller sending Xoff?
;        je     $2802           ;Yes, ignore it.
        mov     bx,out2_head    ;No, advance output que head pointer.
        inc     bx
        cmp     bx,OUTQ_SIZE
        jl      $2302
        xor     bx,bx
$2302:
        xor     ax,ax           ;Assume failure.
        cmp     bx,out2_tail    ;Output que full?
        je      $2902           ;Yes, return failure.
        mov     al,[bp+6]       ;No, get character passed by caller.
        mov     outq2[bx],al    ;Add character to output que.
        mov     out2_head,bx    ;Save new head pointer.
        cli
        call    tickle2_xmit
        sti
$2802:
        mov     ax,1            ;Return success.
$2902:
        pop     bp
        ret
_tty2_out       endp

_tty2_outs      proc    far
;
; Adds an entire null terminated string (supplied by caller) to output que.
; If string is longer than output que, false (0) is returned.
; If there is currently insufficient free space for the entire string to
; be placed on the output que, false (0) is returned.
; If string length is zero, true (1) is returned, but no action takes place.
; Else string is placed on output que and true (1) is returned.
;
```

ttylisr.asm

```
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
        push    si              ;Save callers index registers.
        push    di
        push    ds
        pop     es              ;Pointer ES:DI to callers string.
        mov     di,[bp+6]
        mov     cx,OUTQ_SIZE+2  ;Max legal string length including
        null + 1.
        xor     ax,ax           ;0 = end of string.
        cld                     ;Make sure we increment DI.
        repnz scasb             ;String too long?
        jcxz    $3802           ;Yes.
        sub     cx,OUTQ_SIZE+1  ;No, calculate length of string.
        neg     cx
        je      $3702           ;Zero length string, all done.
$3002:
        mov     ax,out2_tail    ;Calculate free space assuming tail >
head.
        sub     ax,out2_head
        dec     ax
        jge     $3102
        add     ax,OUTQ_SIZE    ;Tail <= head, adjust for que wrap
around.  $3102:
        cmp     ax,cx           ;Enough free space for entire string?
        jl      $3802           ;No.
        mov     si,[bp+6]       ;Yes, set up to xfer string to que.
        lea     di,outq2
        add     di,out2_head
        inc     di
$3202:
        cmp     di,offset dgroup:outq1+OUTQ_SIZE ;Need to wrap pointer
around?
        jl      $3302           ;No.
        lea     di,outq2        ;Yes.
$3302:
        movsb                   ;Transfer one byte to output que.
        loop    $3202
        dec     di
        sub     di,offset dgroup:outq2 ;Calculate new que head
        pointer.
        mov     out2_head,di    ;Save new que head.
        cli
        call    tickle2_xmit
        sti
$3702:
        mov     ax,1            ;Return success.
        jmp     $3902
$3802:
        xor     ax,ax           ;Return failure.
$3902:
```

```
            pop     di
            pop     si
            pop     bp
            ret
;
; Alternate entry for _tty_outs. Queues an array of bytes the size of which
; is specified by the caller.
;
_tty2_outmem:
        push    bp              ;Set up Turbo C call frame.
        mov     bp,sp
        push    si              ;Save callers index registers.
        push    di
        push    ds
        pop     es              ;Initialize ES = DS.
        mov     cx,[bp+6]           ;Get callers byte count.
        or      cx,cx           ;Byte count less than or equal zero?
        jl      $3802           ;Yes, illegal.
        je      $3702           ;Yes, all done.
        cmp     cx,OUTQ_SIZE    ;No, byte count too high?
        jg      $3802           ;Yes.
        jmp     $3002           ;No.
_tty2_outs      endp

_tty2_flush     proc    far
;
; Reset all input and output queue pointers (losing any characters
; currently in either queue). Also resets error counters.
; No arguments or return values.
;
            xor     ax,ax
            cli
            mov     in2_head,ax         ;Reset input que.
            mov     in2_tail,ax
            mov     out2_head,ax        ;Reset output que.
            mov     out2_tail,ax
            mov     _tty2_cnt,ax        ;reset byte count
            mov     xmit2_idle,1
            mov     _tty2_error,ax      ;Reset error counts.
            mov     _tty2_qfull,ax
            mov     send2_xoff,ax       ;Reset Xon/Xoff flags.
            mov     send2_xon,ax
            mov     recv2_pause,ax
            sti
            ret
_tty2_flush     endp

tickle2_xmit    proc    near
        cmp     xmit2_idle,0    ;Transmitter ISR idle?
        je      $5802           ;No.
        mov     dx,_tty2_base   ;Yes, toggle xmit interrupt enable to
        inc     dx              ;force xmit interrupt.
        in      al,dx
        mov     bl,al
        and     al,0FDh
        out     dx,al
        mov     al,bl
        out     dx,al
        mov     xmit2_idle,0
$5802:
        ret
tickle2_xmit    endp

_text   ends
        end
```

merlin.def

```
/*******************************************
*
*    merlin.def  global definition file
*
*******************************************/

#include "merlin.typ"

/*******  The following memory assignments simply map the */
/*******  PC hshi control structures to the MERLIN structures */
/*******  The assignments were copied out of the HSHI reference manual */
/*******  pages 9 and 10.  When used by the code, comments will */
/*******  explain their use.  For more info read the HSHI manual, */
/*******  it is easy to follow and it is short! */

HC_BUF    huge *hc_buf     = (HC_BUF huge *)  (0x00   + MEM_OFFSET);
HR_BUF    huge *hr_buf     = (HR_BUF huge *)  (0x08   + MEM_OFFSET);
HR_RSP    huge *hr_rsp     = (HR_RSP huge *)  (0x0e   + MEM_OFFSET);

SRV_PAR   huge *hc_srv     = (SRV_PAR huge *) (0x10   + MEM_OFFSET);
SPNT      huge *hc_cpos    = (SPNT huge *)    (0x60   + MEM_OFFSET);
float     huge *hc_cvel    = (float huge *)   (0x80   + MEM_OFFSET);
JOINTS    huge *hc_jpos    = (JOINTS huge *)  (0xa0   + MEM_OFFSET);
JOINTS    huge *hc_jvel    = (JOINTS huge *)  (0xc0   + MEM_OFFSET);
LJOINTS   huge *hc_mpos    = (LJOINTS huge *) (0xe0   + MEM_OFFSET);
LJOINTS   huge *hc_mvel    = (LJOINTS huge *) (0x100  + MEM_OFFSET);

SPNT      huge *hr_cpos    = (SPNT huge *)    (0x200  + MEM_OFFSET);
JOINTS    huge *hr_jpos    = (JOINTS huge *)  (0x220  + MEM_OFFSET);
JOINTS    huge *hr_jvel    = (JOINTS huge *)  (0x240  + MEM_OFFSET);
LJOINTS   huge *hr_mpos    = (LJOINTS huge *) (0x260  + MEM_OFFSET);
LJOINTS   huge *hr_mvel    = (LJOINTS huge *) (0x280  + MEM_OFFSET);
LJOINTS   huge *hr_mcyc    = (LJOINTS huge *) (0x2a0  + MEM_OFFSET);

/********* conversions and limits for encoders */
double dtoe[6] =          /* conversion for degrees to encoder cnts */
                                    /* ranges are +- 48000 for j 1,2,3 */
                                    /*          +- 24000 for j 4,6   */
                                    /*          +- 12000 for j 5     */
                                    /* take range / 180 (90 for j 5) for conv */
                  {266.66667,266.66667,266.66667,133.333,133.333,133.333};

double rtoe[6] =          /* conversion for rads to encoder cnts */
                                    /* ranges are +- 48000 for j 1,2,3 */
                                    /*          +- 24000 for j 4,6   */
                                    /*          +- 12000 for j 5     */
                                  /* take range / pi (pi/2 for j 5) for conv */
                {15278.8745,15278.8745,15278.8745,7639.437,7639.437,7639.437};
```

merlin.def

```
long encmin[6] =          /* min encoder reading */
        {-48000,-48000,-48000,-48000,-12000,-48000};
long encmax[6] =          /* max encoder reading */
        {48000,48000,48000,48000,12000,48000};
double degmin[6] =         /* min degrees */
        {-170,-170,-170,-360,-85,-360};
double degmax[6] =         /* max degrees */
        {170,170,170,360,85,360};
double radmin[6] =         /* min rads */
        {-2.967,-2.967,-2.967,-6.283,-1.4835,-6.283};
double radmax[6] =         /* max rads */
        {2.967,2.967,2.967,6.283,1.4835,6.283};

int max_srv_acc[6] = {12,12,12,12,12,20}; /* default servo accel 0..16*/
int max_srv_vel[6] = {8,8,8,8,8,12};       /* default servo veloc 0..32*/
int gain[6] = {4,4,4,4,4,6};              /* default servo gain 1..8*/
double x_pos;                  /* x end tip position */
double y_pos;                  /* y end tip position */
double z_pos;                  /* z end tip position */
double roll;                  /* gripper roll */
double pitch;                  /* gripper pitch */
double yaw;                   /* gripper yaw */
double a[5] = {0,0,17.3,0,0};    /* Denvait Hartenberg parameters */
double alpha[5] = {0,-1.5707,3.14159,-1.5707,-1.7507 };  /* " -90,180,-90,-90 */
double d[5] = {0,0,-12,0,17.25};          /*   "   */
double h[4];                  /* vars used by inverse kin */
double h1_partial;
double theta[6] = {0};

double in_merlin_joint[6];
double gripper_tip[4] = {0,0,-10.5,1};  /* gripper tip x,y,z coordinates */

/* defined w/respect to tool roll */ /* coordinate system */
double ct2,st2,nst2,ct3,nct3,st3,nst3,ct4,nct4,st4,nst4,
          ct5,nct5,st5,nst5,ct6,nct6,st6,nst6,ct7,st7,nst7,
          ct8,nct8,st8,nst8;
                              /* nct7 is not used */
                              /* used in tmat() to allow computation of */
                              /* cos(*tX) & sin(*tX) only once */

double st1_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
```

```
double st2_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },

                      { 0,0,0,1}};

double st3_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double st4_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double st5_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double st6_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double sr1_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double sr2_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double sr3_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};


double wrist_roll;          /* slave wrist roll */
double wrist_flex;          /* slave wrist flex */
double tool_roll;           /* slave tool roll */
                                   /* values for converting optical encoder vals */
                                   /* to radians */
double oe_to_rad[8] = { ENC_RAD, ENC_RAD, UA_ENC_RAD, ENC_RAD,
                           LA_ENC_RAD, ENC_RAD, ENC_RAD, 0 };
/*double hs_lft_deg[8] =    /* fixed, measured hardstop angles */
/*            { 0,0,0,0,0,0,0,0};  */
```

merlin.def

```
int arm_hs_oe[8];     /* hard stop values of the optical encoder */

double exo_l_hs_rad[8] =
                        /* exo left arm joint encoder clockwise hardstop (rads) */
                               { 1.57,   /* shoulder azimuth   */
                                 0.8098, /* shoulder elevation */
                                -0.7662, /* upper arm roll     */
                                 0.785,  /* elbow flex         */
                                -2.12,   /* lower arm roll     */
                                 1.284,  /* wrist radial       */
                                 2.543,  /* wrist flex         */
                                 0.0     /* gripper            */
                               };


double exo_r_hs_rad[8] =
                        /* exo right arm joint encoder clockwise hardstop (rads) */
                               { 2.34,   /* shoulder azimuth   */
                                 0.0349, /* shoulder elevation */
                                 0.0   , /* upper arm roll     */
                                 0.8028, /* elbow flex         */
                                -1.6985,   /* lower arm roll     */
                                 1.284,  /* wrist radial       */
                                -0.8264,  /* wrist flex         */
                                 0.0     /* gripper            */
                               };


double l1 = 6.9375;
double l2 = 5.3075;
double l3 = 5.5156;
double l4 = 4.9375;
double l5 = 12.141;
double l6 = 11.250;

double exo_r_arm[8];
double exo_l_arm[8];

double mt[4][4] =    {{ 0,0,0,0 },
                       { 0,0,0,0 },
                       { 0,0,0,0 },
                       { 0,0,0,1}};

double mt2_0[4][4] = {{ 0,0,0,0 },
                       { 0,0,0,0 },
                       { 0,0,0,0 },
                       { 0,0,0,1}};
```

```
double mt3_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mt4_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mt5_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mt6_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mt7_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mt8_0[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};

double mr6_5[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mr7_5[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mr8_5[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
double mr9_5[4][4] = {{ 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,0 },
                      { 0,0,0,1}};
/*******************
*
*     indexing vars
*
**********************/
double x_offset;        /* indexing offsets */
double y_offset;        /* indexing offsets */
double z_offset;        /* indexing offsets */
int indexing = 0;
double initx_offset = 5; /* initial offsets for work space matching */
```

merlin.def

```
double inity_offset = 0;
double initz_offset = 15;
int lock_wrist = 0;      /* is wrist locked in cal position */
int mba_rqst = 'H';      /* requesting Left Right or H/Both */
int done = 0;            /* gen purpose done flag */
int badcnt = 0;          /* count of bad comm messages */
```

merlin.ref

```
/*******************************************
*
*   merlin.def  global definition file
*
*******************************************/

#include "merlin.typ"

/*******  The following memory assignments simply map the */
/*******  PC hshi control structures to the MERLIN structures */
/*******  The assignments were copied out of the HSHI reference manual */
/*******  pages 9 and 10.  When used by the code, comments will */
/*******  explain their use.  For more info read the HSHI manual, */
/*******  it is easy to follow and it is short! */

extern HC_BUF     huge *hc_buf;
extern HR_BUF     huge *hr_buf;
extern HR_RSP     huge *hr_rsp;

extern SRV_PAR    huge *hc_srv;
extern SPNT       huge *hc_cpos;
extern float      huge *hc_cvel;
extern JOINTS     huge *hc_jpos;
extern JOINTS     huge *hc_jvel;
extern LJOINTS    huge *hc_mpos;
extern LJOINTS    huge *hc_mvel;

extern SPNT       huge *hr_cpos;
extern JOINTS     huge *hr_jpos;
extern JOINTS     huge *hr_jvel;
extern LJOINTS    huge *hr_mpos;
extern LJOINTS    huge *hr_mvel;
extern LJOINTS    huge *hr_mcyc;

/********* conversions and limits for encoders */
extern double dtoe[6];      /* conversion for degrees to encoder cnts */
                                  /* ranges are +- 48000 for j 1,2,3 */
                                  /*          +- 24000 for j 4,6   */
                                  /*          +- 12000 for j 5     */
                                  /* take range / 180 (90 for j 5) for conv */
extern double rtoe[6];      /* conversion for rads to encoder cnts */
                                  /* ranges are +- 48000 for j 1,2,3 */
                                  /*          +- 24000 for j 4,6   */
                                  /*          +- 12000 for j 5     */
                                  /* take range / pi (pi/2 for j 5) for conv */
extern long encmin[6];      /* min encoder reading */
extern long encmax[6];      /* max encoder reading */
```

merlin.ref

```
extern double degmin[6];     /* min degrees */
extern double degmax[6];     /* max degrees */
extern double radmin[6];     /* min rads */
extern double radmax[6];     /* max rads */
extern int max_srv_acc[6];   /* default servo accel 0..16*/
extern int max_srv_vel[6];   /* default servo veloc 0..32*/
extern int gain[6];       /* default servo gain  1..8*/
extern double x_pos;                  /* x end tip position */
extern double y_pos;                  /* y end tip position */
extern double z_pos;                  /* z end tip position */
extern double roll;                 /* gripper roll */
extern double pitch;                 /* gripper pitch */
extern double yaw;                    /* gripper yaw */
extern double a[5];      /* Denvait Hartenberg parameters */
extern double alpha[5];  /* " -90,180,-90,-90 */
extern double d[5];      /*   "   */
extern double h[4];                  /* vars used by inverse kin */
extern double h1_partial;
extern double theta[6];

extern double in_merlin_joint[6];
extern double gripper_tip[4];  /* gripper tip x,y,z coordinates */

/* defined w/respect to tool roll */ /* coordinate system */
extern double ct2,st2,nst2,ct3,nct3,st3,nst3,ct4,nct4,st4,nst4,
          ct5,nct5,st5,nst5,ct6,nct6,st6,nst6,ct7,st7,nst7,
          ct8,nct8,st8,nst8;
                                /* nct7 is not used */
                                /* used in tmat() to allow computation of */
                                /* cos(*tX) & sin(*tX) only once */


extern double st1_0[4][4];
extern double st2_0[4][4];
extern double st3_0[4][4];
extern double st4_0[4][4];
extern double st5_0[4][4];
extern double st6_0[4][4];
extern double sr1_0[4][4];
extern double sr2_0[4][4];
extern double sr3_0[4][4];
extern double wrist_roll;       /* slave wrist roll */
extern double wrist_flex;       /* slave wrist flex */
extern double tool_roll;        /* slave tool roll */
                                /* values for converting optical encoder vals */
                                /* to radians */

extern double oe_to_rad[8];
/* extern double hs_arm_deg[8];  */
extern int arm_hs_oe[8];
                                            /* clockwise hardstop values in radians */
```

merlin.ref

```
extern double exo_r_hs_rad[8];
                                        /* most cw rotation of encoder */
extern double exo_l_hs_rad[8];
                                        /* most cw rotation of encoder */

extern double l1;
extern double l2;
extern double l3;
extern double l4;
extern double l5;
extern double l6;

extern double exo_r_arm[8];
extern double exo_l_arm[8];
extern double mt[4][4];
extern double mt2_0[4][4];
extern double mt3_0[4][4];
extern double mt4_0[4][4];
extern double mt5_0[4][4];
extern double mt6_0[4][4];
extern double mt7_0[4][4];
extern double mt8_0[4][4];
extern double mr6_5[4][4];
extern double mr7_5[4][4];
extern double mr8_5[4][4];
extern double mr9_5[4][4];
/*****************
*
*     indexing vars
*
*********************/
extern double x_offset;        /* indexing offsets */
extern double y_offset;        /* indexing offsets */
extern double z_offset;        /* indexing offsets */
extern int indexing;
extern double initx_offset;  /* initial offsets for work space matching */
extern double inity_offset;
extern double initz_offset;
extern int lock_wrist;      /* is wrist locked in cal position */
extern int mba_rqst;      /* requesting Left Right or H/Both */
extern int done;          /* gen purpose done flag */
extern int badcnt;       /* count of bad comm messages */
```

merlin.typ

```
/****************************************
*
*     This is the type declaration file for Merlin
*
****************************************/

/*    This file declares the data types needed for shared memory */
/*    HSHI operations.  This section is similar to that given    */
/*    in the HSHI Reference Manual, pp. 9 - 10.                  */

#define MBA_PORT 2
#define JR3_PORT 1

#define FF1 59
#define FF2 60
#define FF3 61
#define FF4 62
#define FF5 63
#define FF6 64
#define LA  75
#define RA  77
#define UA  72
#define DA  80
#define FHOME 71
#define FEND 79
#define PI 3.14159
#define DTORAD .017453
typedef struct
           {
               int cyc_clk;
               int cmd_no;
               char command;       /* Switched from HSHI manual */
               char buf_stat;    /* See page 9 */
               int cyc_no;
             } HC_BUF ;

#define   NUM_AXES        6

#define   BUF_HOST       0x55
#define   BUF_HSHI       0xaa
#define   sqr(x)         (x) * (x)

#define   CMD_SET_PARAMS  0x01
#define   CMD_EXIT       0x12
#define   CMD_RD_M_STAT   0x03
#define   CMD_M_POS      0x08
#define   CMD_M_VEL      0x09
#define   CMD_RD_J_STAT   0x07
#define   CMD_J_POS      0x05
#define   CMD_J_VEL      0x06
#define   CMD_RD_C_STAT   0x04
```

merlin.typ

```
#define   CMD_C_POS      0x02
#define   CMD_SET_C_VEL   0x0c
#define   CMD_J_INTERP    0x0d
#define   CMD_C_INTERP    0x0e

#define   MEM_OFFSET   0x0d0000000


typedef struct
          {
             int ech_cmd;
             int ech_cyc;
             int end_cyc;
            } HR_BUF ;

typedef struct
          {
             int rsp_bits;
            } HR_RSP ;

typedef struct
          {
             long max_acc;
             long max_vel;
             long gain;
            } SRVPAR ;

typedef struct
          {
          SRVPAR servo_param[6] ;
          } SRV_PAR ;

typedef struct
          {
             float x;
             float y;
             float z;
             float roll;
             float pitch;
             float yaw;
            } SPNT ;

typedef struct
          {
             float axis[6];
            } JOINTS ;

typedef struct
          {
             long axis[6];
            } LJOINTS ;
```

merlin.typ

```
#define ENC_RAD .001534       /* conversion  encoder cnts to radians */
                                   /* Encoder to Joint Gear Ratio 1:1
                                      360 deg joint rev/ 4096 encoder cnts rev
                                      = .0879 deg per encoder cnt
                                      = .001534 rad per encoder cnt */
#define LA_ENC_RAD .0017858  /* conversion  lower arm roll encoder to rad */ /* Encoder to Joint
                                   Gear Ratio 7.33:1
                                      360 deg joint rev/480 * 7.33 encoder cnts
                                      = .10232 deg per encoder cnt
                                      = .0017858 rad per encoder cnt */
/*#define UA_ENC_RAD .0015098  /* conversion  upper arm roll encoder to rad */
                                   /* Encoder to Joint Gear Ratio 8.67:1 */
                                   /* 360 deg joint rev/480 * 8.67 encoder cnts  */ /* =
                                   .0865 deg per encoder cnt */
                                   /* = .0015098 rad per encoder cnt */
#define UA_ENC_RAD  .00126363  /* 480cnts/encrev  * 10.3642encrev/armrev */ /*
                                     /360deg/armrev = cnts/deg */
                                     /* then convert to rad/cnts */
```

11.0 Appendix C: Bit3 PC-AT Adaptor Card Jumper Settings

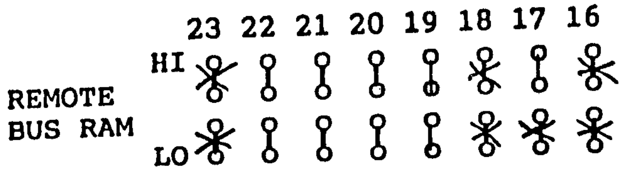
MBA / MerLin
Bit 3 Pc/AT Adaptor
31 Dec 92

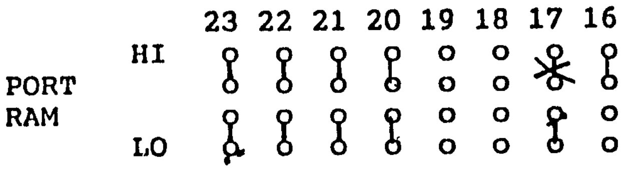
PC-AT ADAPTOR CARD JUMPERS

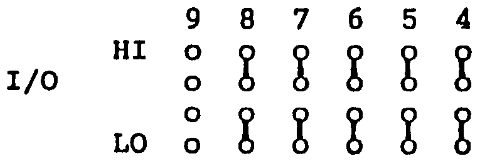
THE SYS JUMPERS (AT CARD)

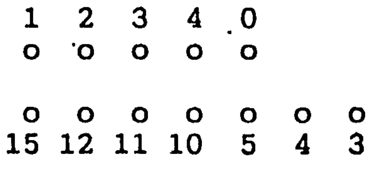
Locate the SYS jumper block at location F2 on the AT board.

```
 SYS
o  o 1   jumper if you want to select byte swap (1)
o——o 2   jumper if you want to select word swap (1)
o  o 3   there should never be a jumper on these pins
o  o 4   jumper if you are using a IBM RT computer
```





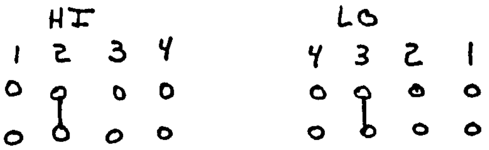




VMEbus INTERRUPTS (0 is the status error interrupt)

PC-AT INTERRUPTS

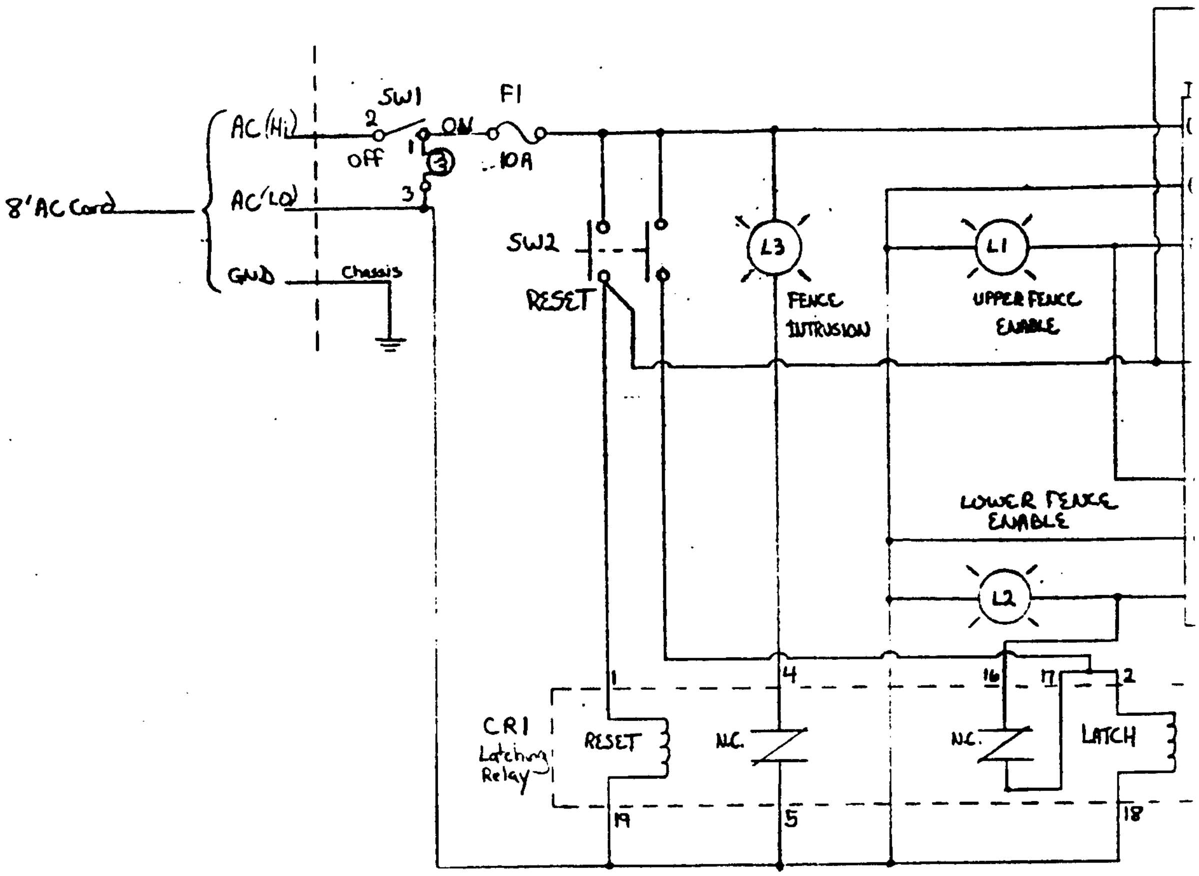
Factory Presets
@ A5/6



12.0 Appendix D: Safety Light Fence Schematics

4 3

D



C

B

A

								PART OR IDENTIFYING NO.	COD
QTY REQD PER ASSY									

PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 3

.XXX = FRACTIONS = ± 1/

.XXXX = BASIC

ALL SURFACES √

MATERIAL

FINISH

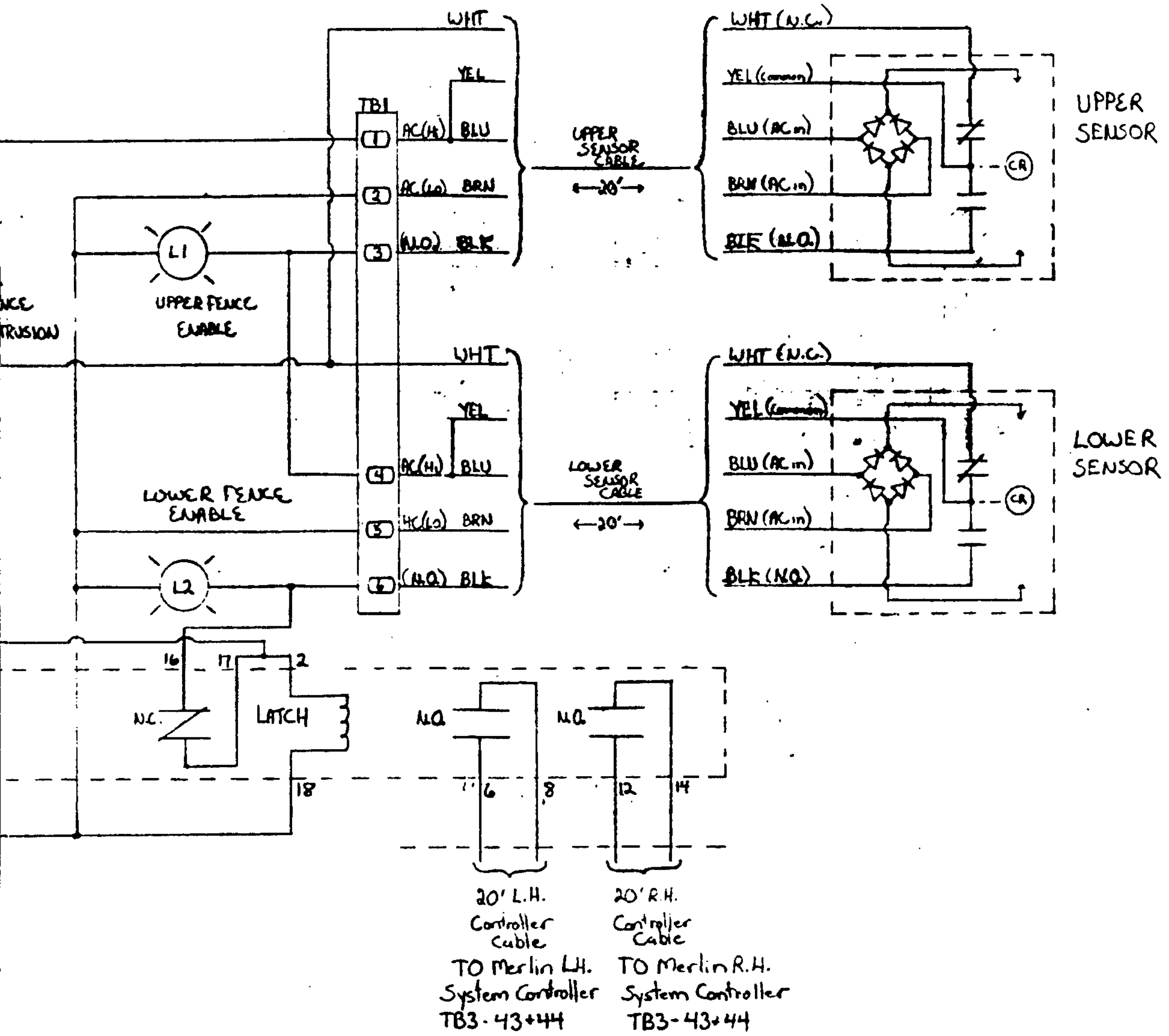
MRC

4 3

2 1

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

D



C

B

		PART OR IDENTIFYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.
		PARTS LIST						

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES	CONTRACT NO.		
TOLERANCES:	DRAWN BY	J Logan	DATE 4-21-88

XX = ANGLES = ± 30'
XXX = FRACTIONS = ± 1/32
XXXX = BASIC
	CHECKED		

	DESIGN		
ALL SURFACES ✓	PROJECT		
MATERIAL	CUSTOMER		
FINISH	QUALITY ASSURANCE		
	MANUFACTURING		

USED ON

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

LIGHT FENCE / MERLIN CONTROLLER SCHEMATICS

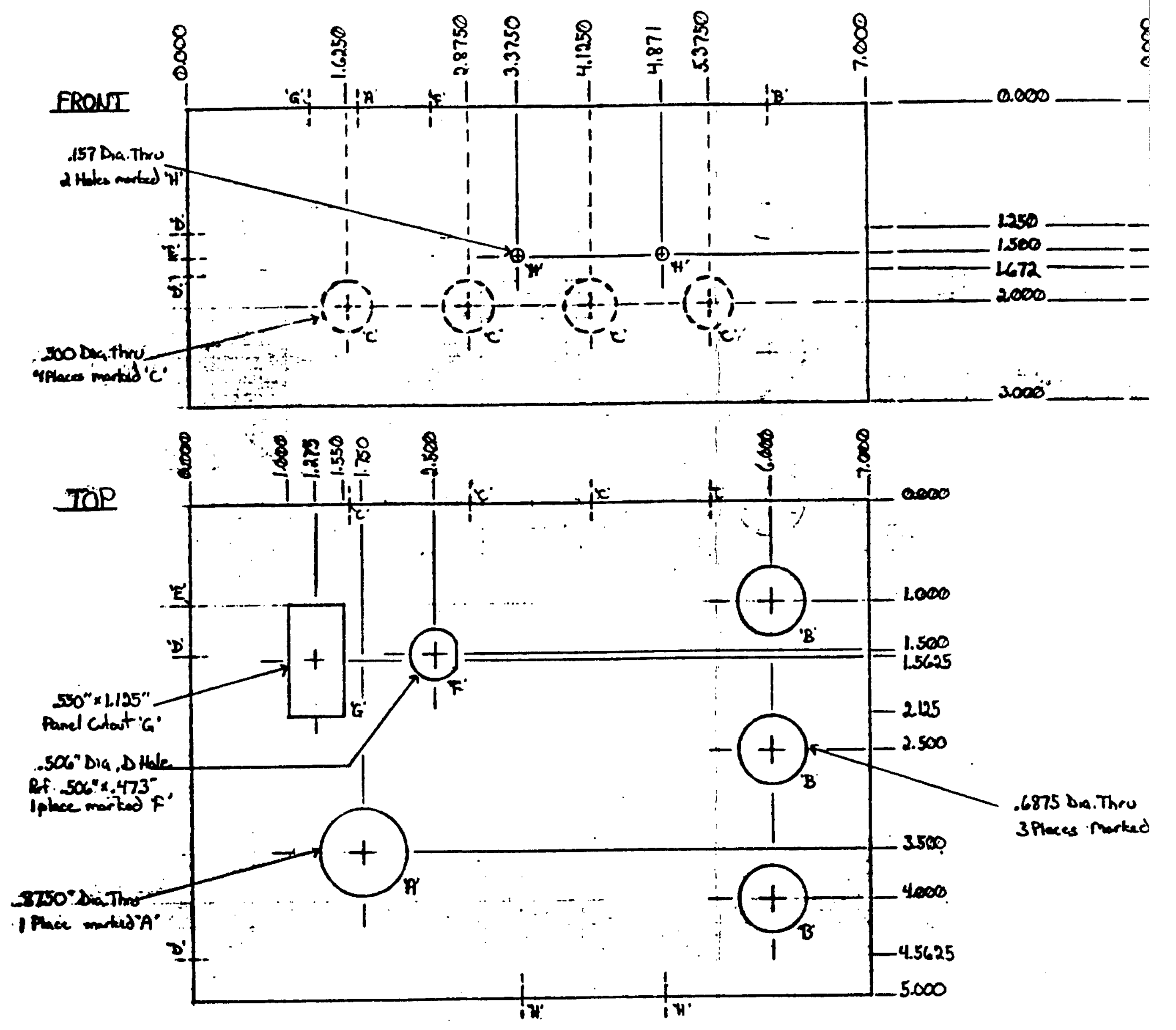
SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590	25 MAY 1988	

SCALE | RELEASE DATE | SHEET OF

A

2 1

2



								PART OR IDENTIFYING NO.	CODE IDENT	NOMEN
QTY REQD PER ASSY										

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES

TOLERANCES:

.XX = ANGLES = ± 30'
.XXX = FRACTIONS = ± 1/32
.XXXX = BASIC

ALL SURFACES ✓

MATERIAL .062 Aluminum

FINISH Clear

CONTRACT NO.

DRAWN BY

CHECKED

DESIGN

PROJECT

CUSTOMER

QUALITY ASSURANCE

MANUFACTURI

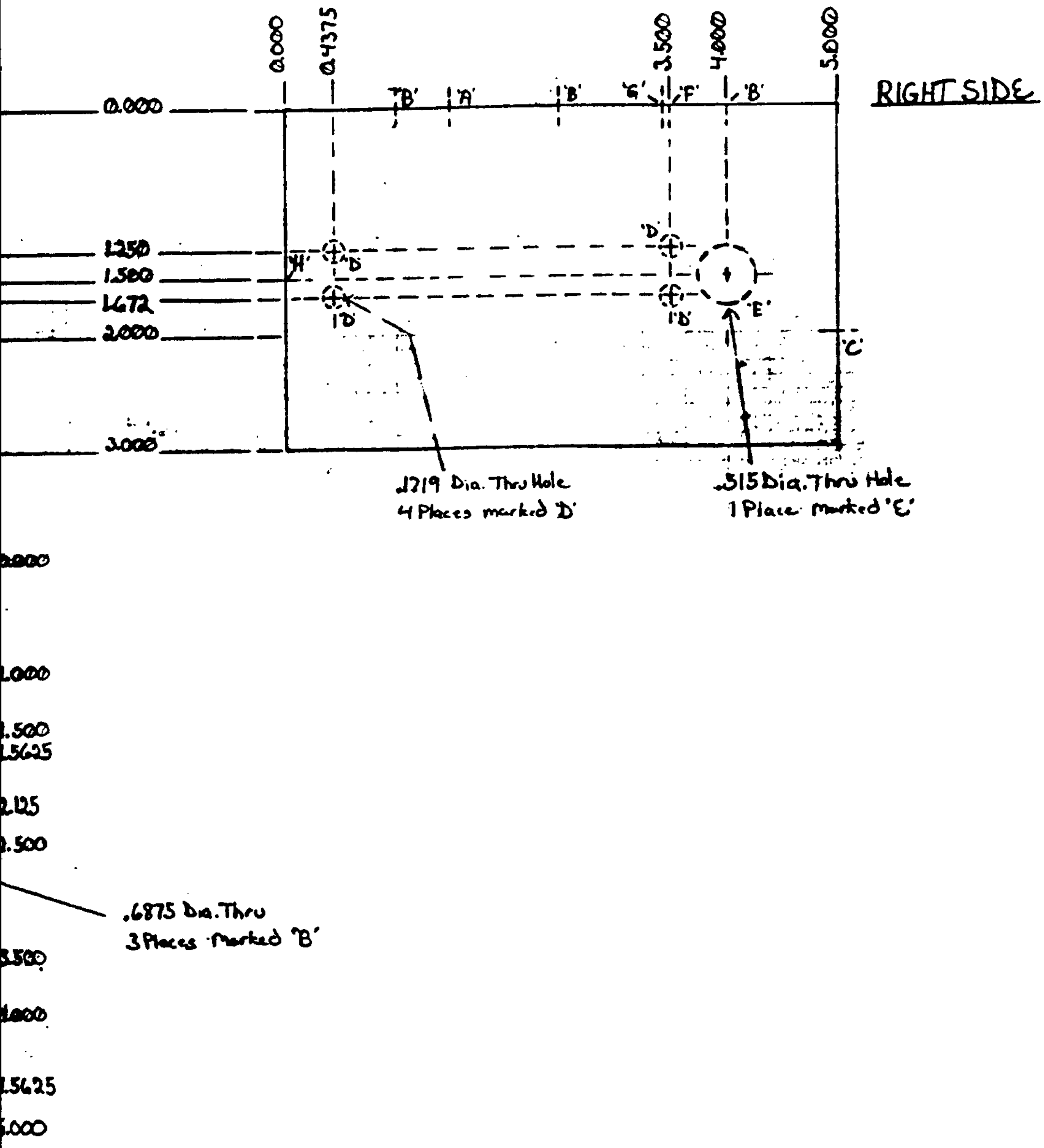
PART NO.	N/A	F/A	NEXT ASSY	USED ON
	QTY REQD		APPLICATION	

MRC

4 3

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED

D



.1219 Dia. Thru Hole
4 Places marked 'D'

.515 Dia. Thru Hole
1 Place marked 'E'

.6875 Dia. Thru
3 Places Marked 'B'

C

B

Chg. 5426-66-27

FYING NO.	CODE IDENT	NOMENCLATURE OR DESCRIPTION	MATERIAL OR MATERIAL CODE	DWG OR SPECIFICATION	ZONE	FIND NO.

PARTS LIST

ERWISE SPECIFIED
S ARE IN INCHES
ERANCES:
ANGLES = ± 30'
FRACTIONS = ± 1/32
URFACES ✓

Aluminum

CONTRACT NO.		
DRAWN BY	J Logan	DATE 4-22-88
CHECKED		
DESIGN		
PROJECT		
CUSTOMER		
QUALITY ASSURANCE		
MANUFACTURING		

SRL SYSTEMS RESEARCH LABORATORIES, INC.
2800 INDIAN RIPPLE ROAD, DAYTON, OHIO 45440

LIGHT FENCE CONTROL BOX

27 APR 1988

A

SIZE	CODE IDENT NO.	DRAWING NO.	REV
C	14590		

SCALE	RELEASE DATE	SHEET OF