Final Report
SOFTWARE DEVELOPMENT TO SUPPORT
SENSOR CONTROL OF ROBOT ARC WELDING
16 April 1985 to 8 May 1986
Contract No. NAS8-36460

ENGINEERING CENTER FOR
AUTOMATED
MANUFACTURING TECHNOLOGY

(NASA-178865) SOFTWARE DEVELOPMENT TO
SUPPORT SENSOR CONTROL OF ROBOT ARC WELDING
Final Report, 16 Apr. 1985 - 8 May 1986
(Clemson Univ.) 173 F HC AC8/MP A01
CSCL 13H G3/37 43502

CLEMSON UNIVERSITY
College of Engineering
Clemson, South Carolina 29631
Final Report

SOFTWARE DEVELOPMENT TO SUPPORT SENSOR CONTROL OF ROBOT ARC WELDING

16 April 1985 to 8 May 1986

Contract No. NAS8-36460

Prepared for:
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

8 May 1986

By
Fred R. Sias, Jr.

Department of Electrical and Computer Engineering
via
Engineering Center for Automated Manufacturing Technology
CLEMSON UNIVERSITY
Clemson, South Carolina 29634-0915
SOFTWARE DEVELOPMENT TO SUPPORT SENSOR CONTROL OF ROBOT ARC WELDING

Abstract......................... 3

1.0 Introduction......................... 4

2.0 Objectives........................... 5
   2.1 Objectives......................... 5
   2.2 Approach....................... 5

3.0 Software Description............... 7
   3.1 Introduction.................... 7
   3.2 Software Overview.............. 8
   3.3 Software Description......... 8

4.0 Description of MINC-23 Interface..... 15
   4.1 MINC Interface Modules........ 15
   4.2 The MINC-Robot Interface..... 15

5.0 Description of Robot Interface....... 18
   5.1 CYRO 750 Hardware Interface... 18
   5.2 Parallel Interface Handshaking Protocol. 18
   5.3 Message Protocol............. 19
   5.4 Message Types.................. 20

6.0 Robot Welding Monitor............... 25
   6.1 Overview..................... 25
   6.2 Problem statement.......... 26
   6.3 Approach.................... 26

7.0 Seam-tracking Monitor............... 27
   7.1 Overview..................... 27
   7.2 Problem Statement............ 27
   7.3 Proposed Approach......... 27

8.0 Conversion to RSX-11M.............. 29
   8.1 Objective..................... 29
   8.2 Features and Limitations of RT-11 29
   8.3 The RSX-11M Operating System. 29
   8.4 Assessment of Operating Systems 31
   8.5 Conversion of CYRO2........... 32

9.0 Conclusions and Recommendations.... 33
   9.1 Conclusions................... 33
   9.2 Recommendations............. 33

APPENDIX.............................. 34
   A. CYRO2 Operating Manual.......... 35
   B. Description of External Device Interface Messages. 42
   C. CYRO2 Software Flow Diagrams.... 55
   D. CYRO2 Program Listings........ 78
ABSTRACT

SOFTWARE DEVELOPMENT TO SUPPORT SENSOR CONTROL OF ROBOT ARC WELDING

Robots are usually used to perform high-volume, repetitive manufacturing tasks. The application of robots in space technology is more associated with low-volume, high-quality-oriented manufacturing. The primary motivation for this effort is to raise the quality and consistancy of certain Gas Tungsten Arc Welds (GTAW) on the Space Shuttle Main Engine (SSME) that are too complex geometrically for conventional automation and are currently done by hand.

Robot welding minimizes the human variability introduced by manual welding methods and improves the overall quality control. In addition, automatic welding of complex shapes can be improved by the use of a positioner table that permits a coordinated movement of the part in relation to the welding torch. Such "coordinated" motion between a robot and a two-axis positioner table is complicated to program and the current on-line methods are awkward. The application of robots in low-volume applications requires improvement in the methods for creating robot control programs.

The use of seam-tracking methods reduces the accuracy with which the seam track must be specified. Instead of using the "teach-and-store" method, it is now possible to create a program automatically from dimensions derived from original Computer Aided Design (CAD) files. A minor mis-alignment of the welding torch due to path approximations derived from design files or metal movement during the weld will be compensated for by the seam-tracking system.

This report documents the development of software for a Digital Equipment Corporation MINC-23 Laboratory Computer to provide functions of a "workcell host" computer for SSME robotic welding. Routines were written to transfer robot programs between the MINC and an Advanced Robotics Cyro 750 welding robot. Other routines provide advanced program editing features while additional software allows communication with a remote CAD system. Access to special robot functions were provided to allow advanced control of weld seam tracking and process control for future development programs.
1.0 INTRODUCTION

Robots are usually used to perform high-volume repetitive manufacturing tasks. The application of robots in space technology is more associated with low-volume, high-quality oriented manufacturing. Robot welding with automatic seam tracking minimizes the human variability introduced by manual welding methods and improves the overall quality control. However, the application of robots in low-volume applications requires improvement in the methods for creating robot control programs.

Normally robot programs are created in a "teach pendant" mode whereby the robot is manually moved through a sequence of movements. Each successive movement is "programmed" by storing the coordinates of the endpoint of the movement along with move velocity and other parameters that may be required to execute a particular manufacturing task. The results of a numerical control step programmed by the "teach" process is converted in the CYRO 750 into a program step in a numerical control "language" similar to standard machine tool languages such as MRP. A number of subprograms can be sequenced together to comprise a complete robot task.

In the case of the CYRO 750 robot attached to the MINC-23, the overall robot task is to perform welds on the space shuttle main engine (SSME). The welding tasks on the SSME consist of a large number of small complex welds that are currently performed manually due to the complexity of the welds. Replacing the manual welds with robot welds is not a simple task.

The conventional robot programming task described above is called "on-line" programming. The robot itself must be dedicated to the programming task which would be particularly time consuming due to the complexity of the weld paths. Weld-path programming will generally take more time than the actual welding process. In addition, automatic welding of complex shapes requires the use of a positioner table that permits a coordinated movement of the part in relation to the welding torch. For example, in welding a cylindrical part it will be necessary to rotate the part rather than simply track the weld so that the weld puddle will remain in a proper orientation relative to gravitational forces on the puddle. Such "coordinated" motion between a robot and a two-axis positioner table is complicated to program at best and the current on-line methods are awkward.

The use of seam-tracking methods reduces the accuracy with which the seam track must be specified. Instead of using the "teach-and-store" method described above, it is now possible to create a program automatically from dimensions derived from original computer based design (CAD) files. A specified mis-alignment of the welding torch due to path approximations derived from design files or metal movement during the weld will be compensated for by the seam-tracking system.

The development of N/C programs for the CYRO 750 robot is further complicated by the fact that the dimensional information is contained in files on a separate Computer Aided Design (CAD) system and the process of converting design files to a robot program is much too complicated to be handled by the small on-line MINC-23 computer attached to the robot. Communication links between two or more computers are necessary and extensive software for automated program creation must be developed.
2.0 OBJECTIVES

2.1 OBJECTIVES

Based on the discussion presented in the Introduction, it is possible to present the overall problem as follows: How to produce programs for welding robots while minimizing interference with research or production operations.

The objectives of this research are twofold. First, a number of software modules have been written to facilitate the creation of Numerical Control programs for the Advanced Robotics Cyro 750 Robot. Included are routines for transferring N/C programs back and forth between the Robot and the MINC-23 computer for storage and retrieval on MINC diskettes. In addition, software has been developed to aid in the off-line programming and editing of N/C programs.

The second area of research are studies designed to aid in planning future applications of the MINC computer. Application areas include communication with a Computer Aided Design (CAD) system so that dimensional information can be passed to the MINC to facilitate automatic development of N/C programs. In addition, further discussions are directed at defining the problems that must be solved to use the MINC computer to record offsets transmitted by the Ohio State seam-tracking system system to facilitate better tracking on multipass welds.

2.2 APPROACH

CAD files containing the SSME dimensional information are generated on a Computer Aided Design (CAD) system. It is anticipated that the conversion of CAD files to N/C robot control programs will be carried out on an intermediate computer such as VAX systems now installed at MSFC. The final link to the MINC-23 or some other computer on-line with the robot can be handled by transferring the N/C programs on floppy diskettes, or over high speed or low speed data communication links. The N/C programs are expected to finally be stored on a computer attached on-line to the robot since the robot is limited to in-memory storage of only nine programs. In a production environment it will be necessary to transfer new programs to the robot as segments of the welding tasks are completed. It is anticipated that the software that is the main subject of this report will continue to be used for that application either on the MINC-23 or a successor digital computer. Figure 2.1 illustrates the overall configuration.
Figure 2.1 Work Cell Host-Robot Configuration.
3.0 SOFTWARE DESCRIPTION

3.1 INTRODUCTION

The software system for MINC-robot communication is called CYRO2. It is a set of routines designed to permit off-line Numerical Control program storage and development on the MINC computer. A previous software system called CYRO1 was primarily a checkout system designed to demonstrate that communication could be established between the MINC computer and the CYRO 750 robot. The CYRO2 system will serve as a basis for developing the functions of a "workcell host computer."

The robot can be programmed in the teach mode described previously or directly from the robot console by typing steps using the CYROVISION numerical control language. The following lines represent an example of a CYROVISION N/C program:

```
N200VA50WA200/ ;Start Welding
N210G4F3/ ;Wait-Arc Stabilize
N220G42C1/ ;Turn Cyrovision On
N230XA23.45YA12.98/ ;Move Robot
N240G43/ ;Turn Cyrovision Off
N260VAOWAO/ ;Stop Welding
N300M02/ ;Program End
```

The above program is comprised simply of strings of text stored as ASCII codes. These are interpreted by the robot controller. The comments are added for clarity and would not appear on the robot control console. When displayed each line of the actual N/C program line is terminated by a "slash". For display purposes the slash replaces an ASCII code for "carriage return" that is actually used to terminate the text string in robot computer memory. The text strings can be manipulated by the standard Digital Equipment Corporation screen editor if the code is modified slightly. Each line of text in the robot is terminated with a carriage return and the editor expects to find each line terminated by a carriage-return code followed by a line-feed code.

To permit off-line development and modification of N/C programs for the robot, it was decided to store program on diskette in a form suitable for manipulation by the standard RT-11 screen editor. All programs are transmitted back and forth between the robot and MINC computer in the formats specified by the External Device interface specifications provide by Advanced Robotics. However, before storage on MINC diskettes, the numerical control programs are modified by the addition of a line-feed code following each carriage-return code.

3.2 SOFTWARE OVERVIEW

CYRO2 is a menu-driven, modular software system that allows the user to create, edit, merge, file on diskette, and transfer N/C control programs back and forth between the MINC computer and the CYRO 750 robot. Also included are utility routines than can be called from the same menu to examine the directory of programs on the diskette, display a directory from the first two lines of programs in the robot, resequence statement numbers...
of an N/C control program, transmit messages to the robot console, list robot operating parameters, display current robot position coordinates, and run a robot N/C program from the MINC console. Only the number corresponding to the desired menu selection and a carriage return need to be typed to execute any given menu function. A menu operation can also be chosen by typing a mnemonic that consists of the letters capitalized in each menu selection. Most of the code is written in FORTRAN to simplify transportation of the system to another computer.

Routines that interact directly with the MINC digital-input and digital-output interfaces are written in PDP-11 assembly language. SEND and RECEIVE (spelled as shown) are two assembler routines that transmit or receive complete arrays of data that are manipulated by calling FORTRAN routines. The two subroutines called SNDBYT and GETBYT could be referred to as device handlers since they handle the actual input and output and take care of the appropriate "handshaking" protocol described in section 5.2 of this report. SEND and RECEIVE transmit complete arrays of data and receive or return appropriate single-byte acknowledge messages.

The assembler routines are the only message-handling subroutines that directly affect or are affected by the interface hardware. At the present time, polling techniques are used to synchronize input/output operations with handshaking flags. The system could be converted to an interrupt-driven environment by merely altering the assembler routines.

Most of the rest of the CYRO2 software package consist of a set of FORTRAN subroutines that prepare or accept various messages that are moved into or out of a 257 word integer array that is, in turn, processed by subroutines SEND and RECEIVE that handle the actual I/O as described above.

3.3 SOFTWARE DESCRIPTION

The mainline program of the CYRO2 package goes by the same name. This routine first initializes the communications linkage and then displays a menu of available selections that correspond to each of the functions that are available in the system. The main menu is shown in Figure 3.1.

Any menu function can be selected by typing either the number or the capitalized letters followed by a carriage return. At that point control is transferred to a subroutine that takes over control and handles the particular function selected. Further interaction with the operator is under control of the selected subroutine and may involve further operator interaction at the CRT keyboard. Control is returned to the main menu when the selected function is completed.

Briefly each of the menu selections function as follows: The REINIT selection establishes a communications link between the robot and external computer. SAVE transfers a program from the robot to a diskette file on the MINC computer. LOAD transfers a program from diskette to the robot. RUN allows any program currently loaded in the robot to be selected by program number. The DIR selection allows the operator to display or print the first two lines of all programs currently loaded into the robot. This constitutes a directory if the first two lines of each program are comments.
CYRO 750 EXTERNAL DEVICE INTERFACE MENU:
1. REINITitialize system (Reset Robot)
2. SAVE robot program to diskette
3. LOAD program from diskette
4. RUN a program
5. DIREctory of programs in robot
6. (DDIR) Diskette DIREctory
7. SHOW a program in the robot
8. LIST a program from diskette
9. EDIT a program on diskette
10. RESequence a program on diskette
11. POSITion of the robot
12. PARAMeters from the robot
13. MESSAGE to the robot console
14. PARK and ignore all messages from robot
15. TRANSfer robot programs from VAX
"E" EXIT to operating system

SELECT>

Figure 3.1. External Device Main Menu
containing program identification. DDIR permits the diskette directory to be displayed or printed. SHOW and LIST display or print programs stored respectively in the robot or on diskette. EDIT transfers control to the RT-11 operating system so that the standard system Editor can be used to create or modify programs. The RES selection resequences the program line numbers of a NC program that has been altered by repeated editing or by the combination of one or more programs. TRANS also returns the operator to the main operating system so that a transfer program can be used to move programs from a CAD computer system to the MINC.

The above menu selections provide the main software features required for off-line program development and modification; however, several additional options are provide since the capability exists in the robot communication package. POSIT and PARAM both request and display robot status data on the MINC console. PARK is a "do nothing" routine that merely discontinues communication between the robot and the MINC while acknowledging any spurious messages from the robot so that error codes are not produced by the robot operating system. The following sections describe each selection in greater detail.

3.4 Subroutine INIT: This subroutine must be executed before any other function is selected. It is automatically invoked whenever the CYR02 software is executed; however, it may be executed from the main menu should robot communications require resynchronization. Following selection of the this function, RESET on the robot teach pendant should be pressed, followed by a wait of two seconds before pressing ENTER or carriage return to return to the main menu.

This subroutine is designed to established communication between the robot and the MINC computer. When the robot is RESET the control software in the CYRO 750 requests device identification and status from any device attached to the external device interface. This subroutine starts by calling subroutine DOUT to clear the parallel output communication register in the MINC computer. (Actually, all bits of the output register are set HIGH due to characteristics of the interface.) Then the subroutine calls the RECEIVE subroutine and waits for a message from the robot. The MINC will wait indefinitely for the robot to transmit a message. When the device identification/status message is received from the robot, the INIT subroutine will respond that the MINC is operational. This is done by sending a type code 129, followed by codes that indicate that the device type is a computer, the device identification is "MNC", that status is operational, and a message for display on the robot console is "MING OK - CYR02 V2.2".

Several messages may be seen on the CRT display reflecting different possible conditions. For example, if no message is ever transmitted by the robot the MINC display will indicate that it is waiting and that enter must be pressed to continue. Other error messages will indicate if the message from the robot was incorrectly received. Next the MINC will try to transmit a reply to the robot up to five times. Finally, the INIT routine will continue to read additional messages and ignore them as there appear to be undocumented additional messages transmitted by the robot following a reset.
3.5 Subroutine SAVE: This routine accepts a program number to be saved on the diskettes from the MINC terminal. The program also asks for a six character name under which the program will be stored. Then a request is transmitted to the robot and the computer receives and files the program returned by the robot. The program is filed using the name typed on the keyboard with the suffix "CYR" appended to it.

For safety, the program also asks if it should overwrite a program with the same name already stored on the diskette. Many possible error messages are contained in the program should any of a number of error conditions are detected during the transfer. The program is a duplicate of the ASCII code contained in the robot.

3.6 Subroutine REQTS: This subroutine is a brief routine that merely sets up an array with the codes appropriate for requesting that one of the programs in robot memory, identified by number, should be transmitted to the external device. It then calls subroutine SEND that handles the actual message transmission.

3.7 Subroutine LOAD: This routine allows the user to select a particular N/C control program to be loaded from the MINC diskette to the robot. Then the program is transmitted to the robot with appropriate handshaking.

The routine requests the name of the diskette file containing the program to be loaded. Only the first six letters need be typed on the keyboard since the routine assumes the "CYR" extension which was appended when the program was stored. Then a message on the MINC CRT display requests the number between 1 and 9 that the N/C control program will be referenced by once it is stored in robot memory. Appropriate error messages are also displayed whenever necessary.

3.8 Subroutine DISKSV: This routine is called by SAVE and handles the actual storage of blocks of data on the MINC diskettes. It merely keeps track of the number of blocks of data transmitted to it by SAVE and fills the last block with NULLS if the block is not completely filled with data. DISKSV locates all of the line-feed characters that serve as program line delimiters and precedes each one with a carriage return. Only the actual program is saved to diskette so that the file will be compatible with the standard RT-11 system editor.

3.9 Subroutine DISKRD: The DISKRD subroutine is just the reverse of the DISKSV subroutine. DISKRD retrieves any N/C control program stored on diskette and strips out the carriage return characters. Then it counts the number of characters in the program and prepares a "header" array that contains the size of the program, the total number of 252 character blocks that will be transmitted, and the actual number of characters stored in the final block of the program if it is not completely filled. Appropriate error messages are displayed if any detected errors occur.

3.10 Subroutine RUN: This routine is used to start any program stored in the robot N/C program memory. It merely accepts a program number from the MINC console keyboard and transmits a message to start that program in the robot. The RUN subroutine has no control over the program in robot memory as these must have been previously loaded from the robot program cartridges.
or by using the LOAD menu selection on the MING. The subroutine displays an appropriate prompt message on the MING CRT prior to accepting a program number from the keyboard. The actual message transmitted by this routine is a "set program mode" message. It must be transmitted while the robot is in a "no activity state", otherwise the message will be ignored.

Following transmission of the RUN message, this routine remains in a state whereby it can receive and acknowledge messages from the robot. Undocumented messages appear to be transmitted by the robot and error conditions will occur if the messages are not acknowledged by the MING. The RUN subroutine will continue to accept messages and ignore them until a key is pressed to return to the main menu.

3.11 Subroutine DIRECT: This routine is used to determine which programs are stored in the robot program memory. This is done by requesting that all programs should be transmitted to the MING from the robot. The DIRECT subroutine then displays a directory that consists of the first two lines of each routine stored in program memory. The first two lines of every N/C control program should contain a program title and other identification information such as descriptive data, dates, and the programmer's name.

3.12 Subroutine DISPLA: The DISPLA subroutine accepts a program number to be displayed from the MING terminal and then sends a request to the robot. The requested program is received and displayed on the MING console CRT or printed on the printer attached to the MING. Selection of output device is under operator control.

Program data received from the robot is formatted as strings of ASCII code alphanumerics terminated by a line-feed code after each program statement. To display a program consistent with the format displayed on the robot console, the line feed is displayed as a right slash (/) followed by a carriage return and line feed that proceeds following program statements.

3.13 Subroutine DDIR: The DDIR subroutine name stands for "Diskette Directory." Selecting this menu option causes the diskette directory to be displayed on the MING CRT or printed in the same format that one finds when the system command "DIR" is typed when no program is executing and the RT-11 system prompt is displayed. The only reason for including this option in the menu is to permit the operator to view the directory without exiting CYRO2 and returning to the operating system.

This routine is basically a direct read of the directory block on the diskette followed by some computations that are necessary to directly access the compact directory information from a FORTRAN program.

3.14 Subroutine LIST: Subroutine LIST is another convenience program to permit listing of any N/C control program stored on diskette with the program name extension ".CYR" without returning to the operating system. To obtain a program listing consistent with robot console displays, the listing routine places a right slash (/) at the end of each program line before the normal carriage return followed by a line feed. The program can be listed on either the MING console CRT or the printer.
3.15 Subroutine RESEQ: This subroutine is used to resequence any N/C control program stored on the MINC diskette. It searches for the "N" followed by a number at the beginning of each line of the program and replaces the current line number with a new sequence number separated by 10 from the proceeding line number. This menu option is used whenever programs have been modified extensively and unused line numbers no longer are available between existing lines to insert new program lines. Since program line numbers are merely used for programmer convenience, this routine simply replaces all the existing numbers with a new sequence starting at 100 and counting by 10's. The subroutine calls another subroutine INSRT that keeps track of the count and converts the line numbers into appropriate ASCII codes.

3.16 Subroutine INSRT: This subroutine is called only by the resequencing subroutine RESEQ. This routine's sole purpose is to keep track of the line number count by tens and convert the line number to a sequence of ASCII codes.

3.17 Subroutine MESSAGE: The MESSAGE subroutine is spelled this way to keep the mnemonic name within the six character name limit. This routine accepts a message from the computer keyboard and transmits it to the robot for display on the robot console display. Appropriate prompt and error messages are generated when necessary.

3.18 Subroutine POSIT: The POSITION subroutine transmits a request for robot positions to the robot with a single byte message set to zero (0). The zero indicates that only one byte per request is desired. The routine then waits for a response from the robot which should include the type code 4 followed by 18 bytes of binary data that convey the current robot position. Each position consists of two sequential bytes: the low-order byte first followed by the high-order byte. The nine coordinates correspond to x, y, and z coordinates in inches, the A and C axis angle in degrees, and each of two positioner x and y axes in degrees.

3.19 Subroutine PARS: This routine requests the current robot parameters and then displays them on the MINC console CRT. It sets up and transmits the request message, receives the returned data, and scales the data before displaying it. Parameters displayed are torch feed rate in inches per minute, wire feed rate in inches per minute, weld level in percent, and AVC/ACC setpoint level in percent. Also the presence of left, right or no oscillation is displayed. The parameters are transmitted once in response to each request.

3.20 Subroutine EDITOR: The EDIT menu selection merely returns control to the operating system so that the operator can use the RT-11 system editor to develop or modify a N/C control program. We could find no way to call this system program without exiting the CYRO2 software system. Since the RT-11 system editor is a very sophisticated and convenient full-screen editor with which most programmers would be familiar, it was decided to use this editor rather than writing a different editor that could be embedded within the CYRO2 package and called directly from the main menu.

Selection of the EDIT option presents a message explaining that control will return to the operating system if desired. Selection of this option
stops the CYR02 system and control returns to RT-11. The RT-11 prompt will appear and the operator should type EDIT <filename.CYR>. The six character filename is some program that exists on the file diskette and the ".CYR" suffix must be appended to the name. If no such file exists the system will ask if you wish to create a new file by that name. The appropriate editor commands should be consulted in the RT-11 system documentation.

Since editing or N/C program creation using the MING computer will usually be an off-line operation, the user will generally not be operating the CYR02 system. Programs created using the RT-11 editor will be in the proper form for use on the robot if the standard syntax is used. The right slash (/) should not be placed in the program file as this is not actually present in N/C control programs. Merely end each program line with a carriage return and the other CYR02 routines will provide proper conversion for display and transmission to the robot. The specific file format used in CYR02 was selected for compatibility with the RT-11 editor.

When an operator wishes to return to an on-line mode, CYR02 must be requested from the RT-11 system prompt and the system INITialized and the robot reset.

3.21 Subroutine PARK: The CYRO 750 robot appears to transmit messages at various times when an external device is on-line. The content of these messages is not apparent from the documentation provided by Advanced Robotics. The problem that occurs is that the robot expects a one byte acknowledgement whenever it transmits a message and the absence of such an acknowledgement causes robot control system to stop and display a timeout error message. The current CYR02 software package is a polled system and is not equipped to handle messages except when specifically in a routine that is programmed to accept messages. To overcome this problem when the CYR02 system is on-line, PARK is an option that places the MING computer in a mode to accept and acknowledge each message but then ignore the content.

The CYR02 communication handlers could be rewritten to function in an interrupt mode to overcome this problem, however, this option has not been implemented since it is not currently clear whether the current hardware will be replaced by another system.

3.22 Subroutine INTER: Control is transferred to this subroutine in response to menu selection TRANS. Like the EDIT selection, this routine merely displays a message and transfers control to the RT-11 operating system so that software for communication with the Intergraph CAD system can be activated.
4.0 DESCRIPTION OF MINC-23 INTERFACE

4.1 MINC INTERFACE MODULES

Digital communications with the MINC-23 computer is relatively straightforward. It is accomplished using standard MINC digital input modules referred to as MNCDI's and MINC digital output modules known as MNCDO's. Up to a maximum of eight of either type may be incorporated into a system. The system referred to as the "External Computer" for the Advanced Robotics system, includes two MNCDI's and two MNCDO's. Each MNCDO or MNCDI includes 16 output and 16 input lines, respectively, plus connections for strobe and reply logic signals. Figure 4.1 shows the connector blocks for the MINC digital input and output modules along with notes that identify the logic signals that may be used to synchronize communication with various peripheral devices.

4.2 THE MING-ROBOT INTERFACE

The communication link between the MINC computer and the Advanced Robotics External Device Interface is shown in Figure 4.2. The MNCDO's and MNCDI's were determined to be suitable for communication with the robot rather than installing an additional Digital Equipment Corporation DRV11 module which would perform essentially the same function. A second pair of digital interface modules is available to perform additional communication duties such as relaying data back and forth between the MINC and an Ohio State University welding seam-tracking system that will be attached to the same robot.

The interface shown in Figure 4.2 includes some features that should be described. Although the MNCDO and MNCDI modules include logic signals for handshaking, it was determined that these inputs and outputs were designed for short-duration pulse signals rather than logic levels that remain set until answered or cleared under program control. Based on the Advanced Robotics External Device Specification, it was decided to use the circuit shown in Figure 4.2 rather than attempt to use the pulse handshaking signals which could be of such a short duration that proper synchronization between the robot and MINC would not be possible.

The interface connections shown in Figure 4.2 utilize the high-order two bits of the digital input and digital output modules for handshaking. Since only 8-bit bytes are transferred back and forth between the two devices, the high-order 8 bits of both the input and output modules are unused for data transfer and may be used for handshaking purposes.

The software described elsewhere was designed to function in a simple polling mode; however, the MNCDI and MNCDO modules make provision for hardware interrupts. An interrupt signal may be obtained either from the pulse logic handshaking inputs or from the high-order two bits on the MINC digital input module. This choice is controlled using software to appropriately set a hardware control/status register in the interface module.
Figure 4.1 MNC Digital Input and Digital Output Units
Figure 4.2 The MINC - ROBOT Parallel Communications Link.
5.0 DESCRIPTION OF ROBOT INTERFACE

5.1 CYRO 750 HARDWARE INTERFACE

Communication with the Advanced Robotics Cyro 750 robot is provided by the addition of a special feature to the basic robot control computer. This interface included both hardware and software and is called the External Device Interface.

The hardware specification provided by the vendor is shown diagramatically in Figure 5.1. Actual communication with an external device is provided by the DRV-11 Parallel Interface Module with the logic signal definitions shown in Figure 5.2. The pin connections and the connector attached to the Cyro 750 is shown in Figure 5.3.

5.2 PARALLEL INTERFACE HANDSHAKING PROTOCOL

The handshaking protocol defined by Advanced Robotics is as follows:

Output from Robot -

1. Write data to port.
2. Set CSR1 (Indicates data ready.)
3. Wait for REQA to be set. (Indicates that external device has accepted data.) If not set in 2000 milliseconds, then exit to error recovery.
5. Wait for REQA to be cleared. If not cleared in 500 milliseconds, then exit to error recovery.

Input to Robot -

The REQB signal line (from the external device) is used to generate an interrupt to indicate that information is available on the port.

1. If REQB is not set, then exit to error recovery.
2. Read data from port.
3. Set CSRO. (Indicates that data has been read.)
4. Wait for REQB to be cleared. If not cleared in 500 milliseconds, then exit to error recovery.
5. Clear CSRO.

The exit to error recovery comments in the above protocol cause program control to return to the mainline program where different consequences will occur depending on the source of the error. In some cases the robot software may attempt to re-transmit a message and in other cases an error message may be displayed indicating that the external device is not ready to communicate with the robot. Table 5.1 shows the various error messages that can be generated.
TABLE 5.1
ERROR MESSAGES

<table>
<thead>
<tr>
<th>ERROR</th>
<th>OCTAL</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1000</td>
<td>LENERR  - Length on input = 0 or 255 error</td>
</tr>
<tr>
<td>201</td>
<td>1001</td>
<td>LRCERR - LRC error on input - after 8 retries</td>
</tr>
<tr>
<td>202</td>
<td>1002</td>
<td>TBBERR - Time between bytes on input too long</td>
</tr>
<tr>
<td>203</td>
<td>1003</td>
<td>THSBER - Time for handshake too long - REQB to go low</td>
</tr>
<tr>
<td>204</td>
<td>1004</td>
<td>THSAHE - Time for handshake on output too long - REQA to go high</td>
</tr>
<tr>
<td>205</td>
<td>1005</td>
<td>THSALE - Time for handshake on output too long - REQA to go low</td>
</tr>
<tr>
<td>206</td>
<td>1006</td>
<td>ADQERR - Free list queue full</td>
</tr>
<tr>
<td>207</td>
<td>1007</td>
<td>TYPERR - Type code not understood</td>
</tr>
<tr>
<td>208</td>
<td>1008</td>
<td>SMGERR - Send message error - not accepted after eight retries</td>
</tr>
<tr>
<td>209</td>
<td>1011</td>
<td>FQEERR - Free list queue empty</td>
</tr>
<tr>
<td>20A</td>
<td>1012</td>
<td>PARERR - Parity error</td>
</tr>
<tr>
<td>20B</td>
<td>1013</td>
<td>DVSERR - Device not operational</td>
</tr>
</tbody>
</table>

5.3 MESSAGE PROTOCOL

Messages to and from the robot are transmitted as variable length sequences of 8-bit bytes. The entire message, along with several control bytes, is transmitted in a single burst. An acknowledge message is expected from the receiving device when the entire message has been received. Handshaking, as described in the previous section, is used to synchronize the transfer of individual bytes of data. A 100 millisecond timeout is established for the initial handshaking response from the receiver and a 1 millisecond timeout is established as the limiting waiting period for a handshaking signal between bytes of data. The following message protocol is quoted from the Advanced Robotics External Device Specification:

"Length - a byte of information is transmitted by the sender indicating the length of the type code and data portion of the message. The length of a message can range from 1 to 254 bytes.

"Sequence Number - a byte identifying each message. This number will be used to reference a particular message, for example, an error message may reference this number to indicate which error caused a message.

"Type Code - a byte indicating the type of message that is being transmitted. This message is used to define the format of the data following, and is application dependent.

"Data - 0 to 253 bytes of information that are application dependent. The number of data bytes plus the type code byte defines the length of the message."
"Longitudinal Redundancy Check - a byte transmitted by the sender to verify that the type code and data were correctly received. This is a software computed check, and is unrelated to any hardware checks that may be performed. The LRC will be computed by exclusive or-ing the length with 'FF', then using the result to exclusive or with the type code, then using the result to exclusive or with each byte of data.

"The message will be complete when a byte is transmitted by the receiver to acknowledge the correct or incorrect receipt of the message from the sender. If the LRC computed by the receiver matches the LRC sent by the sender, then the message was received correctly."

5.4 MESSAGE TYPES

Six message types are supported by the Advanced Robotics External Device Interface. These are described in detail in the APPENDIX. Data is transmitted in 8-bit bytes, however, some data represents 16-bit binary integers, while other data represents ASCII coded text messages. Where data represents a distance, the 16-bit integer is scaled to represent 1/128 inch (.0078) per bit. Angular information is scaled to represent 1/10 degree per bit. The message types specified in the Advanced Robotics External Device Specification (27 April 1984) are as follows:

"Messages from the robot to all devices

- Request Device Identification/Status - Type Code 1
- Program Status Mode - Type Code 2
- Welding Status Mode - Type Code 3
- Robot Positions - Type Code 4
- Special Message to Device - Type Code 5
- Error - Type Code 6
- Robot System Parameters - Type Code 7
- Device Modes - Type Code 8

"Messages from Robot to Sensor Devices

- Sensor Set-up Parameters - Type Code 33
- Sensor Table Parameters - Type Code 34
- Sensor Position Definition - Type Code 35
- Sensor Diagnostic - Type Code 36
- Sensor Calibration - Type Code 37
- Search for seam - Type Code 38

"Messages from Robot to Computer Devices

- Load Program from Computer Acknowledge - Type Code 65
- Save Program to Computer Acknowledge - Type Code 66
- Save Program to Computer - Type Code 67
"Messages from all Devices to the Robot

- Device Identification/Status - Type Code 129
- Set Program Mode - Type Code 130
- Set Welding Mode - Type Code 131
- Request Robot Positions - Type Code 132
- Special Message from Device - Type Code 133
- Error - Type Code 134
- Jog - Type Code 135
- Move Robot - Type Code 136
- Request Robot System Parameters - Type Code 138

"Messages from Sensor Device to Robot

- Override Data - Type Code 161
- In Position Command - Type Code 162

"Messages from Computer Devices to Robot

- Request Save Program to Computer - Type Code 193
- Request Load Program from Computer - Type Code 194
- Load Program from Computer - Type Code 195"
Figure 5.1 External Device Interface Hardware Structure.
Figure 5.2 DRV-11 Parallel Interface.
PARALLEL INTERFACE

**Pin** | **Signal**
--- | ---
2 | IN07
4 | IN06
6 | IN05
8 | IN04
10 | IN03
12 | IN02
14 | IN01
16 | IN00
18 | NEW DATA READY
20 | DATA TRANS
22 | CSR1
24 | CSRO
28 | REQ B
30 | REQ A
34 | OUT07
36 | OUT06
38 | OUT05
40 | OUT04
42 | OUT03
44 | OUT02
46 | OUT01
48 | OUT00
49 | COMMON REFERENCE
50 | COMMON REFERENCE

Connector at Control Cabinet - AMP #208475-1

Mating Connector - AMP #208474-1

Figure 5.3 Parallel Interface Connector
6.0 ROBOT WELDING MONITOR

6.1 OVERVIEW

An important factor related to improvement of weld quality is the ability of a welder to monitor the robot welding operation. Information to be monitored can come from three sources: (1) parameters and control information available from the robot control computer; (2) data available from the vision seam-tracking computer; and (3) data available from other sensors that may be included in the overall welding system.

The software supplied for this contract provides an initial capability to display parameters and position data from the robot control computer. The position of the robot and robot control parameters may be requested by menu selection from the MINC computer console. This program responds only when the data is requested by an operator, however, a more elaborate welding monitor could be written. This software could be expanded to request the data at regular time intervals and provide the operator a continually updated display subject to the limitations of the robot control software to respond to requests.

Offsets and other control information provided by the Ohio State seam tracking machine vision system is not available for display on the MINC computer. This is due to the fact that there is only a single external-device interface on the robot control computer. Either the vision system or the MINC computer can be attached to the robot. Both cannot be attached at one time. The next section of this report will consider possible future enhancements that will allow connection to the vision system through the MINC computer in such a way that offsets and other data passing between the robot and the vision system can be displayed on the MINC console.

Finally, other sensors could provide useful information relative to the progress and status of a weld. These additional devices could be attached to the MINC computer to provide additional data for a status display. Additional interface modules are currently attached to the MINC computer that could be used to provide communication links with other devices such as Penetration Sensors currently being developed by the Rocketdyne Division of Rockwell International. Documentation provided by Rocketdyne shows an IEEE 488 interface that is presumably to communicate with a weld cell host computer. Also additional interface logic cards could be attached to the PDP-11/23 bus in the MINC computer to provide further communication capability.

Quite a broad range of capabilities are possible, however, there are practical constraints and limitations. For instance, the current software package exceeds the memory size of the MINC system and must be loaded a segment at a time using overlays. Monitoring several devices at a time would require several interrupt-driven handlers to be in memory. The current operating system is essentially a single-job monitor, therefore, the various communication tasks would have to be integrated into a single job. Such an expanded system, communicating with several devices, would severely tax the current hardware and is the main reason that a change to a larger computer using a multi-tasking operating system is considered elsewhere in this report.
6.2 PROBLEM STATEMENT

The problem statement based on the above discussion can be stated as follows: How to provide operator welding process displays suitable for monitoring welding operations and status that can be expanded to include data from the robot, the seam-tracking vision system, and other potential sensors; while solving the constraints of limited memory and a single-job operating system on the current MINC-23 computer system.

6.3 APPROACH

If NASA were to specify the data that should be included on an enhanced process display, the present software could be expanded to a certain extent (limited by the single-job operating system) to provide a regularly updated video display on the CRT attached to the MINC computer. Merely providing repeated displays of the data that can currently be requested from the robot is relatively simple. However, if more than one source of information is anticipated in the future, the current software should be converted to a new system with interrupt-driven handlers rather than the current polling-type communications link. Using the MINC computer, some additional features could be included; however, limited memory, the lack of high-speed mass storage disks, and a multi-tasking operating system constrain the additional features that could be added. In addition, each change in the system would require repeated changes to the single-job software, since each additional task would have to be incorporated as part of a single steadily growing program. On the other hand, continually expanding requirements could be readily developed if a larger computer and the RSX-11M operating system were selected for implementation as discussed elsewhere in this report.
7.0 SEAM-TRACKING MONITOR

7.1 OVERVIEW

The previous section of this report mentioned the need to monitor the visual seam-tracking system to provide better process displays. Other factors, however, make it desirable to monitor the seam-tracking system and record offset or correction commands that are transmitted to the robot to correct the tool position while welding a seam. It is anticipated that some welds controlled by the seam-tracking vision system will be multi-pass. That is, the vision system will provide corrections to the torch position during an initial penetration welding pass. Following the initial pass, one or more fill passes will be required on thick material to complete the weld. The additional fill passes will follow the path of the original corrected penetration pass with perhaps some additional offsets to build the fill material. If the original corrections provided by the vision sensor were stored on the external computer (currently the MINC), torch positions during subsequent passes could be simply calculated rather than having the seam-tracking sensor re-track the seam. The offsets to correctly position the fill metal would then be applied to the stored first pass. It is suggested that multi-pass welds would be improved by storing the path of the penetration pass rather than following the seam with the vision system on each pass.

7.2 PROBLEM STATEMENT

This particular problem can be stated as follows: How to develop a system to permit multipass welds to use the coordinates of the initial penetration pass for controlling subsequent fill passes; while at the same time meeting the constraints of a single external device interface on the CYRO 750 robot.

7.3 PROPOSED APPROACH

Based on the constraint of a single external device interface on the robot, the only approach that will allow both the seam-tracking system and the external computer to be attached to the robot together would be for the vision system to pass control information back and forth through the external computer. Fast interrupt handlers would simply accept data from the vision system and immediately pass it to the robot. However, in addition to transmitting offset data to the robot, the intermediate computer would copy the same data into arrays that could then be written to mass storage for later use in recalculating the seam coordinates for subsequent passes. The exact method of determining the seam coordinates is unknown at present, but possibilities are suggested.

An initial idea would be to merely have the robot repeat the original NC program and transmit the stored offsets when appropriate. The possible problem with this approach is the necessity of knowing which segment of the program is being processed when the stored offsets are to be transmitted to the robot. Perhaps the offsets could be time-tagged or somehow related to NC program segments. It is unclear whether the external computer could obtain coordinate information from the robot during the pass at the same time torch correction offsets are being transmitted from the vision system.
It is anticipated that appreciable experimental work would be required to determine whether such a system would work.

It is not likely that the presently configured CYRO2 software package could serve as a basis for a communications link between the vision system and the external device interface on the robot. While the current communications subroutines could serve as models, the handlers for passing data back and forth through the external computer would have to be rewritten. It is anticipated that the MINC-23 computer could be used to test the necessary concepts, however, it probably will be inadequate for a final design. In particular, limited memory, the absence of fast mass storage, and the single-job operating system could be insurmountable obstacles to developing a satisfactory system.

It is recommended that experimental work be initiated to further define the requirements and constraints relative to storage of coordinates or offsets suitable for controlling multi-pass welds.
8.0 CONVERSION TO RSX-11/M

8.1 OBJECTIVE

The objective of the study reported in this section is to investigate the advantages and disadvantages of converting the CYRO2 system from the current operation under the RT-11 operating system to a new configuration under the RSX-11/M real-time operating system.

8.2 FEATURES AND LIMITATIONS OF RT-11

Currently CYRO2 is executed on a MINC-23 general-purpose digital computer under the control of the RT-11 operating system. RT-11 is primarily a small, single-user, real-time operating system designed to function efficiently on small PDP-11 computers. It is designed for interactive program development and dedicated on-line computer operations. The single-job version of RT-11 permits either interactive program development or execution of an online real-time program; however, both operations cannot be carried out at the same time. The RT-11SJ operating system was licensed by the vendor and is currently in use on the MINC-23 at MSFC.

A second version of the RT-11 operating is available that functions in a background/foreground mode. The background/foreground monitor permits program development or other low-priority tasks to execute in the background partition of the system while high-priority foreground tasks operate in a foreground partition. A real-time foreground task may be executed and will have priority access to system resources. The background task will execute whenever the foreground task is not busy.

The resident monitor (RMON) portion of the background/foreground system is twice as large as the RMON component of the single-job system therefore limitations will be placed on the size of programs that can be executed in both background and foreground partitions. The background/foreground operating system is also able to swap segments of the system between main memory and disk backup when suitable disks are available. The MINC-23 has only relatively slow diskette mass storage that would probably be unsuitable for real-time tasks sharing a processor with program development background tasks.

RT-11 is a small efficient real-time operating system that is suitable for operating environments where only a single real-time task must be executed and where program development can take place either at other times or in the low-priority background partition. Interrupt driven processes may be written to provide very fast real-time responses.

8.3 THE RSX-11/M OPERATING SYSTEM

RSX-11/M is the primary real-time operating system for the PDP-11. Quoting the software manual: "It supports multi-tasking, dynamic memory management, multiple programming languages, interactive program development and a wide range of equipment interfaces." RSX-11/M is capable of controlling a number of real-time processes concurrently.
RSX-11/M is a multi-user system. A Monitor Console Routine (MCR) can provide services to a number of terminal users simultaneously while at the same time controlling several real-time tasks. RSX-11/M is the operating system of choice for DEC PDP-11 computers where it is necessary to maintain real-time operations concurrently with software development by several programmers. The real-time response is fast, however, it is slower than the response of a dedicated single-task, event-driven system using an operating system such as RT-11. RSX-11M runs on any PDP-11 processor except the LSI-11, however, "at least 24K of memory is required for concurrent applications execution and program development." RSX-11M is not designed to function in minimum configuration computer systems.

The following material quoted liberally from an PDP-11 Software Manual provides an overview of the capabilities of RSX-11M:

The RSX-11 family of operating systems is designed to provide a resource-sharing environment ideal for multiple real-time activities. The basic facilities that the RSX-11 family provides for handling multiple requests for services while maintaining real-time response to each request are:

> multiprogramming
> priority scheduling
> contingency exits
> power-fail shutdown and auto-restart

In addition, RSX-11M provides:

> disk-based operation
> checkpointing
> optional dynamic memory allocation

The basic unit of work which these operating system facilities service is called the task. A task consists of one or more programs written in a source language such as MACRO ASSEMBLER or FORTRAN, assembled or compiled into an object format, and then built into a task image by the linker utility called the Task Builder. In addition to the normal linkage functions of combining object modules or creating overlays, the Task Builder sets up the basic task attributes that determine the task's resource requirements and relationship to other tasks in the system. The significant task attributes that affect a task's operation in a real-time multiprogramming environment are:

> Partition - the section of memory where the task will reside when it executes.

> Priority - the task's relationship to other tasks competing for system resources.

> Checkpointability - the task's ability to be swapped out of memory when it is not executing to make room for a task of higher priority that is ready to run.
Once a task is built, it can be installed in the system and executed. Task installation simply registers a task's attributes with the system. The task is not in memory, nor is it in competition for system resources. An installed task can be put in active competition for system resources by the operator or by another active task in the system.

When an installed task is activated, the system will allocate necessary resources, bring the task into memory for execution, and place it in competition with other active tasks. Task installation is the basis for efficient task operation. An installed task uses very little memory resource; yet, when the task is needed to service a real-time event, it can be introduced into the system quickly since its basic parameters are already known to the system.

Tasks can also share code and data among themselves through the common partition facility. A common partition is made accessible to the system and to tasks by installing the common partition and the tasks which intend to use it.

The above paragraphs are a text-book description of the features that must be present in a multi-tasking, real-time operating system. RSX-11 is in wide use at NASA and many persons are available to support such a system if CYRO2 is required to operate in a multi-tasking environment.

8.4 ASSESSMENT OF OPERATING SYSTEMS

The selection of an operating system depends on two key elements: the available computer hardware and the anticipated operating environment. Hardware may constrain the choice of operating system. On the other hand, the operating environment may determine the features required of the operating system which, in turn, determine the hardware requirements.

The MINC-23 was originally acquired to support a rather simple interaction with the CYRO 750 robot. It is perfectly feasible to continue to use the present system with the RT-11 operating system to support off-line storage and modification of NC programs for the robot. The required features are available in the CYRO2 software package. However, longer range objectives associated with robot-controlled welding suggest a more complex environment in the future.

The following tasks are anticipated in the use of robots for welding on the space shuttle main engine:

> Continuation of the current task of storing, modifying, and loading numerical control programs for the CYRO 750.

> Recording of computer-vision seam offsets required for multi-pass welds.

> Down loading of initial weld-seam path coordinates from off-line computer aided design (CAD) systems.

> Continued development of software to support the CYRO 750.
The present MINC-23 with the RT-11SJ operating system can handle any one of the above tasks. However, it is anticipated that project growth will require that software development continue concurrently with execution of real-time programs. It would be desirable for a computer to monitor and record seam-following corrections generated by the computer-vision system while concurrently maintaining communications with the CYRO 750 robot as well as downloading seam dimensional information from an off-line CAD system. As projects grow it can usually be anticipated that additional demands will be placed on the computer system.

From this discussion it can be seen that the question of computer operating system is determined simply by the number of concurrent demands that will be placed on the computer. Continuation with RT-11 is to require that future computer operations will be in the single-task environment. If a multi-task environment is anticipated, an early conversion to the RSX-11 operating system would be prudent. Such a conversion would require the use of a larger disk-based computer system to support RSX-11M.

8.5 CONVERSION OF CYRO2

The CYRO2 software is primarily written in FORTRAN with several small assembly language subroutines that handle the actual communications protocol for transmitting data between the MINC-23 and the CYRO 750. Conversion at this time would merely require that the assembly language routines be rewritten to interface with RSX-11M and the new hardware. These assembly language communications routines were specifically written as independent modules to permit easy conversion to another system. The more extensive FORTRAN modules would simply be re-compiled to execute on the new system. The operator interface with the CYRO2 software would remain unchanged.
9.0 CONCLUSIONS AND RECOMMENDATIONS

9.1 CONCLUSIONS

Communications has been established between the CYRO 750 Robot and the Digital Equipment Corporation MINC-23 computer using the digital input-output modules that are part of the MINC-23 system. The interface on the robot is the standard external device interface provided by the manufacturer.

A menu driven set of software routines known as CYRO2 has been written that permit the offline creation, storage, and modification of Numerical Control programs for the CYRO 750 robot. In addition, certain robot operational features that can be controlled or accessed by way of the external device interface have been implemented.

It would be desirable to expand the CYRO2 system to function as a workcell host to handle additional tasks such as serving as a robot-welding monitor and a seam-tracking system monitor.

It will be difficult to expand the MINC-23 system to function as a multitasking work-cell host due to the limitations of the single-job RT-11 operating system.

Some limited additional capability could be developed using the MINC-23 if the CYRO2 communication routines were converted to interrupt-driven handlers.

9.2 RECOMMENDATIONS

1. Convert CYRO2 to operate under control of a multitasking executive such as RT-11/M. Then develop the expanded role as work-cell host by creating additional tasks.

2. If recommendation 1 cannot be implemented, some additional expansion of CYRO2 can be achieved by rewriting the communication handlers to be interrupt driven. NOTE: This approach has serious limitations due to the RT-11 monitor.

3. Expand CYRO2 to include a robot welding monitor.

4. Add a seam-tracking monitor by routing seam-tracking offsets from the Ohio State system through the MINC-23. This route of expansion is limited using the MINC-23 since high-speed mass storage is not available on the system.

5. Include communication with a penetration monitor that can provide additional control capabilities.
APPENDIX

A. CYRO2 OPERATING MANUAL
B. DESCRIPTION OF EXTERNAL DEVICE INTERFACE MESSAGES
C. CYRO2 PROGRAM FLOW DIAGRAMS
D. CYRO2 SOFTWARE LISTINGS
Department of Electrical and Computer Engineering

CLEMSON UNIVERSITY

Clemson, South Carolina 29634-0915

CYRO2 OPERATING MANUAL

8 May 1986

Fred R. Sias, Jr.

Prepared for:
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812
1.0 INTRODUCTION

CYRO2 is a software package designed to execute on a MINC-23 computer manufactured by Digital Equipment Corporation. The MINC-23 is to be attached to the External Device Interface of an Advanced Robotics CYRO 750 robot to provide a capability for off-line manipulation of numerical control programs. This software is a complete revision of CYRO1 which was a simple set of routines written merely to check out the communications interface to the robot.

CYRO2 provides the capability for initializing the communications link, limited control over the robot, as well as provision for saving, editing, loading and executing numerical-control programs. The various software features will be discussed in the next section.

The features of CYRO2 are all controlled from a single main routine with provision for expansion as other capabilities are desired. A main menu allows the user to select any program feature and the menu table can easily be expanded to introduce new features.

2.0 CYRO2 EXECUTION

The CYRO2 software system is executed by the following sequence of operations:

a. Connect the MINC computer to the CYRO 750 EXTERNAL DEVICE INTERFACE using the cable provided. Since there is only one interface plug on the robot, it may be necessary to disconnect the Ohio State vision system. The two systems cannot operate simultaneously without additional hardware modifications.

b. Turn on the robot and load the appropriate system software from a cassette.

c. Turn the communications interface switch on the robot to the "on" position so that the external interface communications link can be established.

d. Place an RT-11 operating system diskette in the left-hand drive which is known as drive DYO. Place the CYRO2 system diskette in the right-hand drive which is referred to as DY1. The operating system is automatically booted when the MINC computer is turned on. The system is initialized and a message will request the current date and time. In addition to the CYRO2 system, drive DY1 will also be used to store all numerical control programs.

e. Execute CYRO2 by typing RUN CYRO2<cr>, where the "<cr>" indicates that you should press a carriage return or newline key.
f. When the prompt appears, initialize the system by pressing "reset" on the robot pendant. Then press <cr> return to the main menu. A message should appear on the robot console to indicate that initialization is complete and communication has been established between the MINC-23 and the robot.

g. The main control menu should appear on the screen as follows:

**GYRO 750 EXTERNAL DEVICE INTERFACE MENU:**

1. REINITialize system (Reset Robot)
2. SAVE robot program to diskette.
3. LOAD program from diskette.
4. RUN a program.
5. DIREc tor y of programs in robot.
6. (DDIR) Diskette directory.
7. SHOW a program in the robot.
8. LIST a program from diskette.
9. EDIT a program on diskette.
10. RESequence a program on diskette.
11. POSITION of the robot.
12. PARAMeters from the robot.
13. MESSage to the robot.
14. PARK and ignore all messages from robot.
15. TRANSfer robot programs from VAX.
"E" EXIT to operating system.

Any item in the menu may be selected either by typing the number or by typing the mnemonic printed in capital letters.

h. Select any desired option from the main menu. The effect of each routine will be described in the next section.

3.0 SOFTWARE DESCRIPTION

Fourteen options are provided that may be selected from the main menu that is used to control the execution of CYRO2. Each of the menu selections basically calls a subroutine that provides the desired capability. All of the subroutines are written in FORTRAN; however, the subroutines may call four assembly-language subroutines that handle the actual communication functions of the software package. The assembly-language routines are the only part of the system that are machine dependant, consequently, the package could easily be transported to another hardware configuration.

The following paragraphs describe the function of each of the main menu selections.
3.1 REINITialize

This selection is called once each time it is necessary to establish communication between the MINC-23 and the Advanced Robotics CYRO 750. Press "reset" on the robot pendant after the REINIT option has been selected. Wait two seconds and press <cr>. Should the robot display an error message associated with the External Device Interface, selecting REINIT and again resetting the robot should clear the error.

Note: Option 14, PARK, has been installed to prevent most errors. See the description of that routine.

3.2 SAVE

The SAVE option is designed to permit the operator to save on diskette any of the 9 numerical control programs that may be currently loaded into the robot memory. The program will ask for the program by number and will also ask the operator for a name under which to file the program on diskette. Up to an eight character name may be selected and the program will append the file suffix ".CYR" when the selected program is transferred to diskette. The operator responses required should be evident from the prompts displayed on the terminal.

3.3 LOAD

The LOAD option is the inverse of the save option. One merely responds to the prompts to select a numerical control program stored on the right-hand diskette drive on the MINC-23. The DDIR option (No. 6) allows one to examine the directory of the diskette to select a program which of course should have a file name that indicates the purpose of the program.

3.4 RUN

The RUN option allows an operator at the MINC-23 console to initiate the execution of any program that is currently loaded into the robot. The desired program is merely selected by number by responding to the prompt on the display. It is not possible for the MINC-23 system to keep track of various programs that may be created in the robot or loaded from various sources such as cassette or diskette. Therefore, the operator is responsible for knowing the application of each program. Option DIR (No. 5) may be used to determine which programs are currently in robot memory as described below.

3.5 DIR

The robot directory option may be selected to determine which numerical control programs are currently stored in the robot memory. This option merely displays the first two lines of each program in robot memory; therefore, each numerical control program should start with two comment lines that contain information
describing the purpose of the program and the name of the person who created the program. It is probably desirable to also include the date the program was created somewhere on the first two comment lines.

3.6 DDIR

The diskette directory option is used to display the directory of any diskette installed in the right-hand drive of the MINC-23 computer. The format is exactly the same as the directory displayed when requested from the operating system prompt. The date that the program was filed is also displayed.

3.7 SHOW

The SHOW option allows one to examine any of the nine numerical control programs currently stored in robot memory. It is selected by number in response to a prompt so the DIR option should be selected first to determine the proper program. The SHOW option also permits the operator to choose to display the program on the CRT or output to the printer for hard copy.

3.8 LIST

The LIST option is analogous to the SHOW option except that programs stored on the diskette may be displayed on the CRT or printed. A prompt will ask for the file name so it is necessary for the operator to know the file name or to obtain the proper name using the DDIR option.

3.9 EDIT

The EDIT option does not invoke a stand-alone editor but instead merely returns the operator to the RT-11 operating system prompt. After receiving that prompt the operator should type "EDIT <File name>" where the file name is any of the names that can be observed on the diskette directory with a suffix ".CYR". When using the system editor the suffix .CYR must be included in the file name. Any numerical control program on the diskette may be edited as long as the standard nc program syntax is correct. The slash (/) found at the end of numerical control programs displayed on the CRT should not be included in programs created with the editor as the slash is not actually part of the program file.

Since use of the RT-11 system editor requires the operator to leave the CYRO2 system, one must issue the "RUN CYRO2" command at the operating system prompt to return to the CYRO2 main menu. It is not necessary to REINITialize the system if the robot has not tried to send External Device Interface messages to the MINC-23 while the editor was in use. Since the robot does send undocumented messages it is possible for the robot to try to send a message and then display an error code when the CYRO2 does not respond. Should this condition ever occur, merely REINITialize the system by selecting option No. 1.
3.10 RESequence

This option allows the line numbers of a numerical control program to be resequenced by tens. In other words, the program numbers originally used will be replaced by lines numbered with the sequence 0010, 0020, 0030, etc. This option is useful when the numbering sequence has been modified by repeated edits and it is necessary to add a line between existing consecutively numbered lines. Memory size limits programs to about 8000 characters.

3.11 POSITION

This option merely displays all robot coordinates at the time the request is made. Only one set of coordinates is displayed although the robot may send additional data. If moving the robot should complete any programmed sequence of actions before returning to the menu.

3.12 PARAMETERS

The PARAM option displays the current values of several operating parameters on the robot. Included are torch feed rate, wire feed rate, percent weld level, AVC/ACC setpoint level, and oscillation status.

3.13 MESSAGE

The MESSAGE option allows one to type a message at the MINC-23 console keyboard that will be displayed at the robot console display. The prompts are self explanatory. A sequence number is requested and sent to the robot but we know of no value for this number. Any number is acceptable.

3.14 PARK

The PARK option permits the MINC-23 to receive and acknowledge any messages from the robot. However, the messages are simply ignored. This option is provided as an alternative to taking the MINC-23 offline so that error conditions will not occur when the robot is being used but is not under the control of the MINC-23.

We believe there are undocumented messages that are transmitted to the external computer that will cause an error condition in the robot if the handshaking on the communication link is not completed. This routine merely provides the appropriate handshaking signals but does nothing with the message. The option is necessary since the MINC-23 must continuously be ready to accept a message to prevent error conditions in the robot. An interrupt driven system would not have this problem but the current software package uses a polling method of establishing the communications link.
3.15 TRANSfer

The TRANSfer option does not execute a program under the control of the CYRO2 system. Instead, this option merely returns control to the operating system where software written by Intergraph personnel may be used for file transfer. This selection was added to CYRO2 by Intergraph personnel and is essentially a slight modification of the software used in the EDIT option.

3.16 EXIT

The EXIT option merely allow the operator to return to the RT-11 operating system when CYRO2 is no longer required.
DESCRIPTION OF EXTERNAL DEVICE INTERFACE MESSAGES

This appendix contains a description of the messages that may be transmitted between the Advanced Robotics CYRO 750 Robot and external computer devices. Messages for communicating between the robot and external sensor devices have not been included. This appendix provides the basis for all of the software described in this report. The following material is a direct quotation from a document dated April 27, 1984, by Advanced Robotics' personnel:

MESSAGE TYPES

There are six message types supported by the external device interface. Distances and angle measurements are referred to in many of the messages in these different message types. For consistency, the following scale factors will be used when referring to distances and angles:

- Distances: 1/128 inch (0.0078) per bit
- Angles: 1/10 degree per bit

MESSAGE CONTENTS - ROBOT TO ALL DEVICES

1. **Request Device Identification/Status** - is a message sent at reset time requesting the device identification and hardware status of the device. The result of the request will be a Device Identification/Status message from the device, indicating the existence, software and hardware version numbers, and the status of the hardware that can be determined by the device.

   Coding - Type Code = 1

2. **Program Status Mode** - indicates to the device that the specified N/C program has been started or stopped.

   Coding - Type Code = 2
   Status (one byte):
   - Program Started = 1
   - Program Stopped = 2
   Program Number (one byte - 1 to 9)
3. **Welding Status Mode** - indicates to the device that welding has been started or stopped by the N/C program.

   **Coding** - Type Code = 3
   Status (one byte):
   Welding Started = 1
   Welding Stopped = 2

4. **Robot Positions** - indicates to the device what the current robot positions are.

   **Coding** - Type Code = 4
   - X axis position - inches (two bytes, low byte, then high byte transmitted)
   - Y axis position - inches (as above)
   - Z axis position - inches (as above)
   - A axis position - degrees (two bytes, low byte, then high byte transmitted)
   - C axis position - degrees (as above)
   - X axis position - C positioner - degrees (as above)
   - Y axis position - C positioner - degrees (as above)
   - X axis position - D positioner - degrees (as above)
   - Y axis position - D positioner - degrees (as above)

5. **Special Message to Device** - is a message that will pass ASCII data that is placed in a corresponding N/C command to the device. This message is envisioned to allow special features of some devices to be enabled without the need to change robot software. It may also be used to send information messages from the N/C program to the device. There is a corresponding message from the device to the robot that will display on the operator's terminal.

   **Coding** - Type Code 5
   Variable number of ASCII bytes (up to 253) to be interpreted by the device for special function operation.

6. **Error** - is a message indicating that an error has occurred in the robot control, and what that error is. The device will be required to make a decision based on the error as to the proper course of action to take.

   **Coding** - Type Code 6
   Error Number - to be defined as needed
7. **Robot System Parameters** - is a message indicating tha a robot system parameter has changed. Some of the system parameters will be torch, feedrate, welding level, wirefeed speed, and left and right oscillations.

**Coding** - Type Code 7

- Torch Feedrate - inches per minute (two bytes, low byte then high byte transmitted)
- Wirefeed Feedrate - as above
- Weld Level - percent of power supply output (two bytes, low byte, then high byte transmitted - one bit equals 0.1 percent)
- AVC/ACC - Setpoint Level - weld level setpoint as defined in the N/C program for Automated Voltage Control and Automatic Current Control (two bytes, low byte, then high byte transmitted - one bit equals 0.1 percent)
- Oscillation - indicates that a left or right oscillation has occurred (one byte):
  - None = 0
  - Left Oscillation = 1
  - Right Oscillation = 2

8 **Device Modes** - is a message telling the device whether the messages being received by the robot will be executed or not. For example, this will tell a sensor when it should start sending override data, or a host computer that a safety switch has been released, and that it has control of the robot.

**Coding** - Type Code 8

- Device Type (one byte):
  - Sensor Device = 1
  - Computer Device = 2
- Device Identification - three characters as defined in the Device Identification/Status message.
- Device Status (one byte):
  - Device On = 1
  - Device Off = 2
MESSAGE CONTENTS - ROBOT TO COMPUTER DEVICE

1. **Load Program from Computer** Acknowledge - a reply message from the robot to indicate whether the robot is ready to load a program from the computer or not. The robot will accept a program from the computer device only when it is in a no activity state and a keyswitch on the robot controller is in the device enable position. The no activity state is when no program is running or no mode is being executed. The robot goes into a no activity mode after a halt, a stop program, or the program finishes. It will remain in this mode until the operator initiates a function from the pendant, or a program is remotely started. This message is also used while loading a program from the computer device as an acknowledge between program blocks.

   **Coding** - Type Code 65
   Acknowledge (one byte):
   - Not ready to load = 0 (try again later)
   - Ready to Load = 1
   Program Number - one byte (1 - 9). This is the program number sent in the Request to Load Program message, and is used for verification purposes.

2. **Save Program to Computer** Acknowledge - a reply message from the robot to indicate whether the robot is ready to save a program to the computer or not. The robot will send a program to the computer device only when it is in a no activity state. The robot goes into a no activity mode after a halt, a stop program, or the program finishes. It will remain in this mode until the operator initiates a function from the pendant, or a program is remotely started.

   **Coding** - Type Code 66
   Acknowledge (one byte):
   - Not Ready to Save = 0
   - Ready to Save = 1
   Program Number - one byte (1 - 9). This is the program number sent in the Request to Save Program message, and is used for verification purposes.
   Program Size - two bytes, low byte send first indicating the length of the program (number of bytes) about to be sent to the computer.

3. **Save Program to Computer** - is a message to the computer device that contains the N/C program requested. This message is transmitted the same as other messages with the addition of a block number in the data portion of the message. The length of the N/C program is sent to the computer device in the acknowledge message, and the save program message may be sent multiple times to save the entire program. Between blocks, the robot will expect
a request to save program message, indicating that the robot is to send the next block of the program.

Coding - Type Code 67
Block Number - a byte starting with zero, incremented by one for each segment until the complete program has been sent to the computer.
N/C Program - variable number of data bytes.

MESSAGE CONTENTS - ALL DEVICES TO ROBOT

1 Device Identification/Status - is a message sent as a response to a Request Device Identification/Status. This message will indicate existence, software and hardware version numbers, and the status of the hardware that can be determined by the device.

Coding - Type Code = 129
Device Type (one byte): 
  Sensor Device = 1
  Computer Device = 2
  Both = 3
Device Identification - three bytes (ASCII characters) that identify the device. These characters will be used in the N/C program to reference this device.
System Status (one byte):
  Not Operational = 0
  Operational = 1
Variable number of ASCII characters defining the program version level and other information. This information is printed on the operators terminal.

2 Set Program Mode - indicates to the robot that an N/C program of the specified number is to be started or stopped. This message will be executed only if the keyswitch for the device is in the enable position and the robot is in a no activity mode. If a program is running and the program start message is received, then it will be ignored. If the robot is in a no activity state and the program stop message is received, it will be ignored. The program stop message will place the robot in the no activity state if the keyswitch is enabled.

Coding - Type Code 130
Status (one byte):
  Program Start = 1
  Program Stop = 2
Program Number (one byte - 1 to 9)
3 Set Welding Mode - indicates to the robot that welding is to be enabled or disabled. This function is identical to pushing the torch enable on the robot control pendant, however, it is controlled by an external device. In each weld, the welding will be enabled if the torch enable on the robot control pendant is pushed until a Disable Welding message is received by the robot. If a Disable Welding message is received, then an Enable Welding message, and the robot is still in the welding portion of the N/C program, then the arc will be re-established.

Coding - Type Code = 131
Status (one byte):
  Enable Welding = 1
  Disable Welding = 2

4 Request Robot Positions - indicates to the robot that the current robot positions are to be sent to the device. This command can initiate the transfer of robot positions at a periodic rate of up to 10 Hertz or only when requested.

Coding - Type Code 132
Rate (one byte) - 0 to 10 times per second. A rate of 0 will indicate to robot to only send current robot positions once, until requested again.

5 Special Message to Robot - is a message that will pass ASCII data to the robot and onto the operator's terminal. This message is envisioned to allow special messages of some devices to be transferred to the operator's terminal without the need to change the robot software. There is a corresponding message from the robot to the device that will allow enabling of special features in the device.

Coding - Type Code = 133
Variable number of ASCII bytes (up to 253) to be displayed on the operator's terminal.

6 Error - is a message indicating that an error has occurred in the device and what the error is. The device will be required to make a decision based on the error as to the proper course of action to take.

Coding - Type Code 134
Error Number - one byte (defined as needed)
Error Messages - variable number of ASCII bytes defining the error. This message will be sent to the operator's terminal.

7 Jog - is a message that initiates a jog function of the robot manipulator axes. Once initiated, the axes will continue to move at the specified rate until a Stop Jog command is received.
This command can only be executed if the robot is in a no activity state. That is, when no program is running or no mode is being executed. The robot goes into a no activity mode after a halt, a stop program, or the program finishess. It will remain in this mode until the operator initiates a function from the pendant, or a program is remotely started. Note that there are some special N/C commands that place the robot in a special no activity mode (M1 and M3). When these commands execute, a halt is executed. In order to proceed, the operator must push the RUN button on the robot control pendant, or the external device must send a Set Program Mode - Program Start Message.

Coding - Type Code 135
X axis rate - one byte 0 - 100 percent (one bit equals one percent)
Y axis rate - as above
Z axis rate - as above
A axis rate - as above
C axis rate - as above

8 Stop Jog - is a message that will stop the jog motion of the robot manipulator as specified.

No = 0
Yes = 1

Coding - Type Code 136
All axes - (One byte)
X axis
Y axis
Z axis
A axis
C axis

9 Move Robot - is a message telling the robot to move to an absolute position in the robot coordinate system. Due to the overhead for communications, the frequency at which this command can be sent is unknown. This message, just as the jog message, must be sent when the robot is in a no activity state.

Coding - Type Code 137
X axis position - inches (two bytes, low byte, then high byte transmitted)
Y axis position - as above
Z axis position - as above
A axis position - as above
C axis position - as above

10 Request Robot System Parameters - indicates to the robot that the current robot system parameters are to be sent to the device. The robot will respond with a Robot System Parameters message (Type code 7)

Coding - Type Code 138
MESSAGE CONTENTS - COMPUTER DEVICES TO ROBOT

1 Request Save Program to Computer - is a message requesting that the robot save the specified program to the computer device. The robot will respond with a Save Program to Computer Acknowledge message indicating whether the robot is ready to send the program to the computer and if so, what the program size is in bytes. The robot will then send the program to the computer a message at a time. The computer must respond between save program messages with this request message to indicate that it is ready for the next program block. (Note: The first Save Program to Computer message containing a block of program data follows immediately after the Save Program to Computer Acknowledge message without waiting for another Request Save Program to Computer message.)

Coding - Type Code 193
Program Number - one byte 1 - 9

2 Request Load Program from Computer - is a message requesting that the robot load the program of specified size and number from the computer device. The robot will respond with a Load Program from Computer Acknowledge message indicating whether the robot is ready to load the program from the computer or not.

Coding - Type Code 194
Program Number - one byte 1-9
Program Size - number of characters (bytes) in program (two bytes - low byte first, high byte second)

3 Load Program from Computer - is a message from the computer device that contains the N/C program. This message is transmitted the same as other messages with the addition of a block number in the data portion of the message. The length of the N/C program is sent to the robot in the Request Load Program from Computer message, and the load Program message may be sent multiple times to load the entire program. Between blocks, the robot will respond with a load program from computer acknowledge message to indicate that the next block is to be sent.

Coding - Type Code 195
Block Number - a byte starting with zero, incremented by one for each segment until the complete program has been sent to the computer.
N/C Program - variable number of data bytes
APPENDIX C

PROGRAM FLOW DIAGRAMS
INITIALIZE SYSTEM

DISPLAY MENU

SELECT:
REINIT
SAVE
LOAD
RUN
DIRECTORY
DISKETTE DIR.
SHOW ROBOT PROG.
LIST DISK PROG.
EDITOR
RESEQUENCE
ROBOT POSITION
PARAMETERS
MESSAGE
PARK
INTERGRAPH
EXIT

EXIT?

YES
STOP

NO
INIT

CLEAR OUTPUT REG.

WAIT FOR MESSG.

ERROR CODES?

YES → 1

NO

SETUP MESSAGE

TRANSMIT TO ROBOT

ACKNOWLEDGED

YES → 2

NO

5 TRIES?

YES

PRINT GIVEUP MSG.

RETURN
TRANSMIT REQUEST TO SAVE MESSAGE

ERROR CODES

ACKNOWLEDGEMENT

RECEIVE PROGRAM AND FILE ON DISKETTE

ERROR CODES

SUCCESSFUL FILE MESSAGE

RETURN
PRINT ERROR MESSAGES

TRY AGAIN?

RETURN TO POINT TO REPEAT
LOAD

ACCEPT PROG. NAME TO BE LOADED.

ACCEPT NO. TO BE ASSIGNED IN ROBOT.

RETRIEVE FILE AND DETERMINE SIZE.

TRANSMIT REQUEST-TO-LOAD TO ROBOT.*

READY TO LOAD?

YES

READ DISK FILE INTO ARRAY.

TRANSMIT DATA BLOCKS

LOAD ACKNOWLEDGE?

PRINT SUCCESSFUL LOAD MESSAGE.

RETURN

* Error checking on all communications.
* NOTE: All messages transmitted between the robot and external computer involve handshaking and extensive error checking. For simplification, this is not shown in most flow diagrams.
RUN

ACCEPT PROG. NO. FROM KEYBOARD

SET UP MESSAGE

SEND IT

ANY ERRORS?

YES

RECEIVE ACKNOWLEDGEMENT

ANY ERRORS?

YES

DISPLAY CRT MESS'G. TO OPERATOR

PRINT ERROR MESSAGES

RECEIVE ACKNOWLEDGEMENT AND IGNORE OTHER MESS'GS

RETURN
DIRECT

TRANSMIT REQUEST TO SAVE *

RECEIVE ACKNOWLEDGE *

IF NOT FIRST BLOCK
TRANSMIT REQ. TO SAVE *

RECEIVE BLOCK *

DISPLAY FIRST TWO LINES OF PROGRAM

END OF PROG. ?

YES

NINE PROG'S. REC.

YES

RETURN

* Error checkin on all communications.
DDIR

HOUSEKEEPING

SELECT PRINTER OR CRT OUTPUT

DIRECT ACCESS *
DIRECTORY BLOCK

DECODE DIRECTORY DATA

PRINT DISKETTE DIRECTORY AND FILE DATES

RETURN

* Displays error messages if errors occur during access.
NO

DISPLA

ACCEPT PROGRAM NO. TO DISPLAY

CRT OR PRINTER?

TRANSMIT REQUEST TO SAVE *

RECEIVE ACKNOWLEDGE *

DISPLAY PROGRAM NO.

RECEIVE BLOCKS OF DATA *

OUTPUT BLOCK OF DATA

CHANGE CR'S TO "/"

END OF PROGRAM?

NO

YES

RETURN

* Error checking on all communications.
LIST

INPUT FILE NAME

SELECT PRINTER OR CRT OUTPUT

CONVERT TO RAD50

GET CHANNEL AND OPEN FILE *

DISPLAY PROG. NAME

READ TO END OF FILE

CLOSE & FREE CHAN.

CALC. PROG. LENGTH

DISPLAY OR PRINT PROGRAM

RETURN

* Displays error messages if file not found.
EDITOR

DISPLAYS MESSAGE

RETURN TO THE OPERATING SYSTEM?

YES

EXITS TO OPERATING SYSTEM TO USE SYSTEM EDITOR

NO

RETURN

STOP
RESEQ

REQUEST FILE NAME

INPUT FILE NAME

CONVERT TO RAD50

GET CHANNEL AND OPEN FILE

READ PROG TO ARRAY

CLOSE FILE AND FREE CHANNEL

CALL INSRT AND RENUMBER LINES

REWRITE TO DISKETTE

RETURN
INSRT
INCREMENT LINE NUMBER COUNT
CONVERT LINE NO. TO ASCII INTEGERS
REPLACE OLD LINE NO. WITH NEW NO.
RETURN
MESSAGE

DISPLAY MESSAGE REQUEST

INPUT MESSAGE

CALCULATE MESSAGE LENGTH

BUILD MESSAGE ARRAY

TRANSMIT TO ROBOT

ANY ERRORS?

YES

DISPLAY ERROR MESSAGES

NO

RETURN
PARK

DISPLAY CRT MESSAGE

CALLS RECEIVED TO ACCEPT MESSAGE FROM ROBOT

ERROR PARAMETER SET TO ONE? NO

YES

RETURN
INTER

DISPLAYS MESSAGE

RETURN TO THE OPERATING SYSTEM?

YES

EXITS TO OPERATING SYSTEM TO USE COMMUNICATION PROG'S.

NO

RETURN

STOP
SEND

SET UP ARRAY LENGTH COUNTER

CALCULATE CHECKSUM

PLACE CHECKSUM IN ARRAY

GET A BYTE FROM ARRAY AND CALL SNDBYT

NO

ARRAY SENT?

YES

WAIT FOR ACK BYTE

CHECKSUM ERROR?

RET. ERROR PARAM.

TIMEOUT ERROR?

RET. TIMEOUT PARAM.

RETURN
YES

SET TIMEOUT PARAM.

RETURN

HOUSEKEEPING

REC. MESSG. LENGTH

UPDATE CHECKSUM

REC. SEQ. NUMBER

UPDATE CHECKSUM

A
RECEIVE BYTE
STORE IN ARRAY
UPDATE CHECKSUM

END OF DATA?

YES
RECEIVE CHECKSUM

AGREE WITH CALCULATED CHECKSUM

YES
TRANSMIT CONFIRMATION MSG. REC'D TO ROBOT

NO
SET CHECKSUM ERROR PARAMETER FOR CALLING PROGRAM

TRANSMIT CHECKSUM ERROR CODE TO ROBOT

RETURN
SNDBYT

HOUSEKEEPING

LATCH DATA IN OUTPUT REGISTER

SET REQB TO INDICATED THAT DATA IS READY

INITIALIZE TWO-BYTE TIMEOUT COUNTER

CHECK FOR HANDSHAKING

ACKNOWLEDGED?

YES

CLEAR REQB DATA READY FLAG

NO

TIMEOUT?

YES

CLEAN UP STACK

ERROR RETURN

NO

CSRO CLEAR?

YES

CLEAN UP STACK

MOVE THE RETURN ADDRESS FORWARD BY TWO WORDS

NORMAL RETURN
READ A BYTE AND SAVE ON STACK

SET REQA TO INDICATE BYTE REC'D

CSR1 RESET?

CLEAR REQA

POP DATA INTO R0

RETURN
DOUT

HOUSEKEEPING

PUT DATA IN R0

MASK DATA

COMPLEMENT DATA

LATCH DATA TO INITIALIZE OUTPUT REGISTER

RETURN
APPENDIX D

CYRO2 PROGRAM LISTINGS
PROGRAM CYR02

PURPOSE: MAINLINE PROGRAM FOR COMMUNICATING WITH ADVANCED ROBOTICS EXTERNAL DEVICE INTERFACE OPTION.

INPUT: USER INTERACTION FROM MINC KEYBOARD.

PROCESSING: THIS PROGRAM MERELY CHECKS FOR INPUT ERRORS AND CALLS SELECTED SUBROUTINES THAT HANDLE ALL PROCESSING.

OUTPUT: DISPLAYS CYRO 750 EXTERNAL DEVICE INTERFACE TEST MENU:

1. REINITialize System. (Reset Robot!)
2. SAVE robot program to diskette
3. LOAD program from diskette
4. RUN a program
5. DIrectory of programs in robot
6. (DDIR) Diskette Directory
7. SHOW a program on robot
8. LIST a program on diskette
9. EDIT a program
10. RESequence a program on diskette
11. POSITION of Robot
12. PARAMETERS from Robot
13. MESSAGE to Robot Console
14. PARK and ignore all messages from Robot.
15. TRANSFER robot program for vax

'E' EXIT to the operating system

CALLED BY: THIS IS THE MAINLINE PROGRAM.

CALLS TO: INIT,SAVE,LOAD,Run,DIRECT,DISPLAY,DDIR,LIST,
RESEQ,POSIT,PARS,EDITOR,INTER,XCOM (or TESTER), DOUT.

SPECIAL INTERFACE REQUIREMENTS: COMMUNICATION SUBROUTINES CALLED THAT USE THE MINC DIGITAL INPUT/OUTPUT MODULES.

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
5-18-84 FRED R. SIAS, JR. (803)656-3375/SYSTEM DESIGN
5-31-85 FRED R. SIAS, JR. (803)656-3375/NEW NARRATION
6-02-85 FRS DELETE IN-POSITION
7-05-85 FRS & JKM FINISHED DEBUGGING!
7-11-85 FRS CHANGED TO CYR02
9-6-85 FRS ADDED "PARK"
1-10-86 DAVID STILES INTERGRAPH 3-2100: VAX FILE TRANSFER
5-07-86 FRS SET UP AUTO INITIALIZE ON ENTRY
DECLARATIONS

REAL*8 INPUT

INTEGER*2 IDATA(257), RT, WT

INTEGER*2 IARRAY(2500)

BYTE ARRAY(5000), BPDATA(520)

INTEGER*2 IARRAY(2500), IPDATA(260)

COMMON /INOUT/ ARRAY, BPDATA

EQUIVALENCE (ARRAY, IARRAY)

EQUIVALENCE (BPDATA, IPDATA)

INITIALIZATION

WT=7

RT=5

WRITE (WT, 100)

FORMAT('!', ' CYRO2 - V2.2', ' M/M/NC-23 COMMUNICATIONS SOFTWARE FOR', ' CYRO 750 EXTERNAL DEVICE INTERFACE', ' F. R. SIAS, JR.', ' ECE DEPT. - CLEMSON UNIVERSITY', ' CLEMSON, SOUTH CAROLINA 29634-0915', '(803) 656-3375', ' 6 MAY 1986', ' TO CONTINUE PRESS RETURN>

READ (RT, 110) X

FORMAT(G6.0)

AUTO INITIALIZE CALLUP ON PROGRAM ENTRY

CALL INIT

MAIN SELECTION MENU

CONTINUE

WRITE (WT, 310)

FORMAT('!', ' CYRO 750 EXTERNAL DEVICE INTERFACE MENU', ' 1. REINITIALIZE system (Reset Robot!)', ' 2. SAVE robot program to diskette', ' 3. LOAD program from diskette and verify', ' 4. RUN a program', ' 5. DIRECTORY of programs in robot', ' 6. (DDIR) Diskette Directory', ' 7. SHOW a program in the robot', ' 8. LIST a program from diskette', ' 9. EDIT a program on diskette', ' 10. RESEQUENCE a program on diskette')
POSITio of the robot',
PARAMeters from the robot',
MESSAGE to the robot',
PARK and ignore all messages from robot',
TRANSfer robot program from VAX',
'E' EXIT to operating system',
SELECT>'$')
C
ACCEPt SELECTION AND BRANCH TO APPROPRIATE SUBROUTINE

0020 READ (RT,400) INPUT
0021 400 FORMAT(A8)

0022 IF (INPUT.EQ.'1') CALL INIT
0024 IF (INPUT.EQ.'REINIT') CALL INIT
0026 IF (INPUT.EQ.'2') CALL SAVE
0028 IF (INPUT.EQ.'SAVE') CALL SAVE
0030 IF (INPUT.EQ.'3') CALL LOAD
0032 IF (INPUT.EQ.'LOAD') CALL LOAD
0034 IF (INPUT.EQ.'4') CALL RUN
0036 IF (INPUT.EQ.'RUN') CALL RUN
0038 IF (INPUT.EQ.'5') CALL DIRECT
0040 IF (INPUT.EQ.'DIR') CALL DIRECT
0042 IF (INPUT.EQ.'7') CALL DISPLA
0044 IF (INPUT.EQ.'SHOW') CALL DISPLA
0046 IF (INPUT.EQ.'6') CALL DDIR
0048 IF (INPUT.EQ.'DDIR') CALL DDIR
0050 IF (INPUT.EQ.'8') CALL LIST
0052 IF (INPUT.EQ.'LIST') CALL LIST
0054 IF (INPUT.EQ.'9') CALL EDITOR
0056 IF (INPUT.EQ.'EDIT') CALL EDITOR
0058 IF (INPUT.EQ.'10') CALL RESEQ
0060 IF (INPUT.EQ.'RES') CALL RESEQ
0062 IF (INPUT.EQ.'11') CALL POSIT
0064 IF (INPUT.EQ.'POSIT') CALL POSIT
0066 IF (INPUT.EQ.'12') CALL PARS
0068 IF (INPUT.EQ.'13') CALL MESSAGE
0070 IF (INPUT.EQ.'MESSAGE') CALL MESSAGE
0072 IF (INPUT.EQ.'PARK') CALL PARK
0074 IF (INPUT.EQ.'14') CALL PARK
0076 IF (INPUT.EQ.'PARAM') CALL PARS
0078 IF (INPUT.EQ.'15') CALL INTER
0080 IF (INPUT.EQ.'TRANS') CALL INTER
0082 IF (INPUT.EQ.'E') STOP
0084 IF (INPUT.EQ.'EXIT') STOP

0086 GOTO 300

0087 END
FORTRAN IV     Storage Map for Program Unit CYR02

Local Variables, .PSECT #DATA, Size = 001022 ( 265. words)

Name   Type Offset   Name   Type Offset   Name   Type Offset
INPUT   R*8 001002   RT   I*2 001012   WT   I*2 001014
X       R*4 001016

COMMON Block /INOUT /, Size = 012620 ( 2760. words)

Name   Type Offset   Name   Type Offset   Name   Type Offset
ARRAY   L*1 000000 Eav BPDATA L*1 011610 Eav IARRAY I*2 000000 Eav
IPDATA I*2 011610 Eav

Local and COMMON Arrays:

Name   Type Section Offset ----Size----- Dimensions
ARRAY   L*1  INOUT  000000  011610 ( 2500.) (5000)
BPDATA  L*1  INOUT  011610  001010 ( 260.) (520)
IARRAY  I*2  INOUT  000000  011610 ( 2500.) (250)
IDATA   I*2  $DATA  000000  001002 ( 257.) (257)
IPDATA  I*2  INOUT  011610  001010 ( 260.) (260)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name   Type Name   Type Name   Type Name   Type Name   Type
DDIR   R*4 DIRECT R*4 DISPLA R*4 EDITOR R*4 INIT I*2
INTER  I*2 LIST I*2 LOAD I*2 MEGAGE I*2 PARK R*4
PARS   R*4 POSIT R*4 RESEQ R*4 RUN R*4 SAVE R*4
**FORTRAN IV**

```
C*********************************************************************************************
C
C MODULE NAME: SUBROUTINE INIT

C PURPOSE:
SUCCESSFUL COMPLETION OF THIS SUBROUTINE ESTABLISHES THAT
COMMUNICATION PROTOCOLS ARE WORKING PROPERLY.

C INPUT:
THIS SUBROUTINE IS REQUESTED FROM THE MINC CONSOLE
PRIOR TO RESETTING THE ROBOT. RESET ROBOT, WAIT TWO
SECONDS AND THEN PRESS ENTER TO RETURN TO MENU.

C PROCESSING:
WHEN THE ROBOT IS RESET IT WILL REQUEST A DEVICE IDENT/STATUS
FROM THE MINC COMPUTER.
TYPE CODE = 1

THIS SUBROUTINE WILL RESPOND THAT THE MINC IS OPERATIONAL
AND WILL TRANSMIT THE APPROPRIATE CODES.

C OUTPUT:
CODES TRANSMITTED TO ROBOT:
  TYPE CODE = 129
  DEVICE TYPE (1 BYTE) = 2 (MEANS THAT COMPUTER DEVICE)
  DEVICE IDENT, (3 ASCII BYTES) = "MNC"
  SYSTEM STATUS (1 BYTE) = 1 (MEANS OPERATIONAL)
  ASCII MESSG = "MINC OK - CYR01 V1.0"

C CALLED BY: CYR02

C CALLS TO: DOUT, SEND, RECEIVE

THIS SUBROUTINE WILL CALL DOUT TO CLEAR OUTPUT REGISTER,
THEN IT CALLS THE RECEIVE SUBROUTINE TO WAIT
FOR THE INITIAL MESSAGE FROM THE ROBOT.
THE MINC WILL WAIT INDEFINITLY FOR THE ROBOT TO RESPOND.
TO QUIT WAITING HIT ANY KEY ON THE MINC
CONSOLE AND THE PROGRAM WILL RETURN TO THE MAIN MENU.
SEND SUBROUTINE CALLED TO TRANSMIT MESSAGE TO ROBOT CONSOLE.

C SPECIAL INTERFACE REQUIREMENTS:
  MINC DIGITAL I/O MODULES
  CYRO 750 EXTERNAL DEVICE
  INTERFACE OPTION

C REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)
C DATE       PROGRAMMER                CONTACT/ACTION/REMARKS=====
C 7-25-84     FRED R. SIAS, JR. (803)-656-3375/DESIGN & PROGRAMMED
C 5-31-85     FRED R. SIAS, JR. (803)-656-3375/NEW NARRATION

C*********************************************************************************************
```
INTEGER*2 IDATA(257),RT,WT

BYTE INPUT

RT=5
WT=7

FIRST OUTPUT 000000 TO CLEAR DIGITAL OUTPUT REGISTER

IUNIT=0
IMASK='000000

CALL DOUT(IUNIT,IMASK,IERR,IMASK)

CONTINUE

WRITE (WT,50)
FORMAT('1'/////,' Press RESET on ROBOT, then wait two seconds,','1
   ///,' then press RETURN key on MINC to return to menu,','1
   2
   ////////////,' Press RETURN> ';)

CALL RECEVE(IERROR,ILEN,ICSUM,IDATA)

TIME OUT ERROR

IF (IERROR.EQ.1) GOTO 199.

CHECKSUM ERROR IF IERROR = 2
IF (IERROR.EQ.2) GOTO 2000

ILLEGAL ERROR CODE IF GREATER THAN 2
IF (IERROR.GE.3) GOTO 3000

GOTO 5100

HERE IF TIMES OUT AT ANY TIME

GOTO 5100

HERE IF TIMES OUT AT ANY TIME

WRITE (WT,200)
FORMAT('///,' NO INITIALIZATION MESSAGE FROM ROBOT','///,' PRESS *RETURN* TO CONTINUE> ';)

READ (RT,5230) INPUT

RETURN
HERE TO PROCESS CHECKSUM ERROR
CONTINUE
WRITE (WT, 2010)
FORMAT(//, 'CHECKSUM DOES NOT COMPUTE!',//,
1 'TRY AGAIN? (Y OR N)> ', $)
READ (RT, 5230) INPUT
IF (INPUT.EQ. 'Y') GOTO 10
RETURN
HERE TO PROCESS ILLEGAL ERROR CODE AND RETURN TO MENU
CONTINUE
WRITE (WT, 3010)
FORMAT(//, 'RECEIVED AN ILLEGAL ERROR CODE FROM RECEIVE',
1 'SUBROUTINE.',//,
2 'PRESS 'RETURN' TO CONTINUE> ', $)
READ (RT, 5230) INPUT
RETURN
CONTINUE
HERE TO PROCESS ACCEPTABLE REQUEST
RETURN STATUS/IDENTIFICATION INFO TO ROBOT
IDATA(3)=129
DEVICE TYPE = 1 I.E. THIS IS A COMPUTER
IDATA(4)=1
DEVICE IDENTIFICATION = 'MNC'
IDATA(5)=77
IDATA(6)=78
IDATA(7)=67
SYSTEM STATUS IS OPERATIONAL
IDATA(8)=1
SEND MESSAGE 'MNC OK - CYR02 V2.2'
IDATA(9)=77
IDATA(10)=73
IDATA(11)=78
IDATA(12)=67
IDATA(13)=32
IDATA(14)=79
IDATA(15)=75
IDATA(16)=32
IDATA(17)=45
IDATA(18)=32
CYR02
IDATA(19)=67
IDATA(20)=89
IDATA(21)=82
IDATA(22)=79
IDATA(23)=50
IDATA(24)=32
C
IDATA(25)=86
IDATA(26)=50
IDATA(27)=46
IDATA(28)=50
C
IDATA(1)=26
C
ILEN=28.
C
WILL TRY TO SEND FIVE TIMES
C
DO 5200 I=1,5
C
CALL SEND(IERROR,ILEN,IDATA)
C
IF (IERROR.EQ.0) GOTO 5500
C
5200 CONTINUE
C
WRITE (WT,5210)
5210 FORMAT(/A' ' MINC GAVE UP AFTER 5 TRIES TO SEND!'/)
C
WRITE (WT,5220)
5220 FORMAT(/A' ' PUSH *RETURN* TO CONTINUE> '/)
C
READ (RT,5230) INPUT
5230 FORMAT(A4)
C
RETURN
C
5500 CONTINUE
C-----------------------------------------------------------------GET SECOND MESSAGE
C
CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)
C
TIME OUT ERROR
0080 IF (IERROR.EQ.1) GOTO 199
C
CHECKSUM ERROR IF IERROR = 2
0082 IF (IERROR.EQ.2) GOTO 2000
C
ILLEGAL ERROR CODE IF GREATER THAN 2
0084 IF (IERROR.GE.3) GOTO 3000
C-----------------------------------------------------------------KEEP READING MESSAGES
0086 DO 7000 I=1,25
C
0087 CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)
C
TIMEOUT ERROR
0088 IF (IERROR.EQ.1) GOTO 6000
C
C CHECKSUM ERROR IF IERROR = 2
0090 IF (IERROR.EQ.2) GOTO 2000
C C ILLEGAL ERROR CODE IF GREATER THAN 2
0092 IF (IERROR.GE.3) GOTO 3000
C
0094 7000 CONTINUE
C
C ---------------------------------------------
C C HERE IF INITIALIZATION COMPLETE AND OK
0095 WRITE (WT,5510)
0096 5510 FORMAT(//' ',/,' INITIALIZATION COMPLETE AND OK!','//

1       ' PRESS 'RETURN' TO CONTINUE > ','$
0097 READ (RT,5230) INPUT
C
0098 RETURN
C
0099 6000 CONTINUE
C
0100 WRITE (WT,6010)
0101 6010 FORMAT(//' ',/,' INITIALIZATION COMPLETE AND OK!','//

1       ' NO MORE MESSAGES FROM ROBOT','//
1       ' PRESS 'RETURN' TO CONTINUE > ','$
0102 READ (RT,5230) INPUT
C
0103 RETURN
0104 END
### Local Variables, .PSECT $DATA, Size = 001026 (267 words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I$2</td>
<td>001024</td>
<td>IERROR</td>
<td>I$2</td>
<td>001016</td>
<td>IERR</td>
<td>I$2</td>
<td>001014</td>
</tr>
<tr>
<td>IERROR</td>
<td>I$2</td>
<td>001016</td>
<td>ILEN</td>
<td>I$2</td>
<td>001020</td>
<td>IMASK</td>
<td>I$2</td>
<td>001012</td>
</tr>
<tr>
<td>INPUT</td>
<td>L$1</td>
<td>001006</td>
<td>IUNIT</td>
<td>I$2</td>
<td>001010</td>
<td>RT</td>
<td>I$2</td>
<td>001002</td>
</tr>
<tr>
<td>WT</td>
<td>I$2</td>
<td>001004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I$2</td>
<td>$DATA</td>
<td>000000</td>
<td>001002</td>
<td>(257)</td>
</tr>
</tbody>
</table>

### Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUT</td>
<td>R$4</td>
<td>RECEIVE</td>
<td>R$4</td>
<td>SEND</td>
<td>R$4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE SAVE

PURPOSE:

This routine accepts a program number to be saved from the terminal, then a request is transmitted to the robot and the computer receives and files the program returned to it.

INPUT:

Program requests robot program number from user at MINC CRT. Then request a six character file name to be used when filing on diskette on MINC.

PROCESSING:

HANDSHAKING:

Computer transmits Type Code 193 - REQUEST TO SAVE PROGRAM TO COMPUTER.
The computer waits for Type Code 66 - SAVE PROGRAM TO COMPUTER ACKNOWLEDGE.
The robot then transmits the first block of data (block 0) after a brief pause.
Then computer accepts and files a sequence of blocks using Type Code 193 to indicate when it is ready for each block.
Each block of data transmitted to the robot has a header containing the Type Code 67 followed by a sequential block number and the actual N/C program data.

OUTPUT:

Writes program on MINC diskette using name accepted from MINC keyboard. *.CYR* suffix added by this routine.

CALLED BY: CYR02

CALLS TO: REQTS, IGETC, ICLOSE, IFREEC, IRAD50, SCOPY, RECEIVE, INSERT, DISKSV

SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULE

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
6-28-84 FRED R. SIAS, JR. (803)656-3375/SYSTEM DESIGN/PROGRAMMED
6-01-84 FRED R. SIAS, JR. (803)656-3375/REVISED NARRATION
7-22-85 J. KEITH MCELVEEN (803)656-3375/REVISED NARRATION

************************************************************
INTEGER*2 NUMBER, IDATA(257), RT, WT, FILNO, HEADER(3)
BYTE LERROR, INPUT, FILNAM(15), NAMFIL(7)
REAL*8 FILE

CALL SCOPY('DY1 CYR', FILNAM)
DATA FILNAM(13) /00/
DATA NAMFIL(7) /00/

WRITE (WT, 10)
FORMAT(///)' Input NUMBER of Program in robot to be saved,',
1 '/ ' To EXIT, press RETURN.'
*,)$
READ (RT, 110) NUMBER
FORMAT(I2)

IF(NUMBER.EQ.0) RETURN
IF(NUMBER.LT.1, OR. NUMBER.GT.9) GOTO 400
CONTINUE
WRITE (WT, 115)
FORMAT(/A' Type NAME of Program to be saved on diskette? ',")
READ (RT, 119) (NAMFIL, I=1,6)
FORMAT(A61)

IF (NAMFIL(.EQ.'')) RETURN
CALL INSERT(NAMFIL, FILNAM(4),6)
CALL IRAD50(12, FILNAM, FILE)
ICHAN=IGETC()
IF (ICHAN.LT.0) STOP 'NO CHANNEL'
IERROR=IFETCH(FILE)
IF (IERROR.NE.0) STOP 'BAD FETCH'
IERROR=LOOKUP(ICHAN, FILE)
IF (IERROR.EQ.-2) GOTO 150
WRITE (WT, 120)
FORMAT(' File already exists! OVERWRITE IT? (Y or N) ')
READ (RT, 130) INPUT
FORMAT(A1)
IF (INPUT.EQ.'Y') GOTO 150
CALL ICLOSE(ICHAN, IERROR)
IF (IERROR.LT.0) STOP 'ERROR CLOSING CHANNEL'
CALL IFREEC(ICHAN)
GOTO 114
0048 150 CONTINUE

C
0049 CALL ICLOSE(ICHAN, IERROR)
0050 IF(IERROR.LT.0) STOP 'ERROR CLOSING CHANNEL'
0052 CALL IFREEC(ICHAN)

C
0053 160 CONTINUE

C
C NOW TRANSMIT A 'REQUEST TO SAVE' TO ROBOT
C
0054 CALL REQTS(NUMBER, IERROR)

C
0055 IF (IERROR.EQ.1) GOTO 2100
0057 IF (IERROR.EQ.2) GOTO 2200
0059 IF (IERROR.GE.3) GOTO 2300

C
C NOW WAIT FOR SAVE ACKNOWLEDGE - TYPE CODE 66
C
0061 200 CONTINUE

C
0062 CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)

C
0063 IF (IERROR.EQ.1) GOTO 3100
0065 IF (IERROR.EQ.2) GOTO 3200
0067 IF (IERROR.GE.3) GOTO 3300

C
0069 ITYPE=IDATA(3)
0070 IACK=IDATA(4)
0071 IPNUM=IDATA(5)
0072 ISIZE=IDATA(6)+256*IDATA(7)
C C WRONG MESSAGE TYPE RECEIVED
C 0073 IF (ITYPE.NE.66) GOTO 1100
C 0075 IF (IPNUM.NE.NUMBER) GOTO 1200
C C NOT READY TO RECEIVE CODE = 0
C 0077 IF (IACK.EQ.0) GOTO 1300
0079 IF (IACK.GE.2) GOTO 1000
C 0081 IBLKCT=0
C C SENDS BLOCK NUMBER AND 252 BYTES OF DATA IN EACH BLOCK
C 0082 ITOTBK=ISIZE/252
0083 IREM=ISIZE-ITOTBK*252
0084 IF (IREM.NE.0) ITOTBK=ITOTBK+1
C 0086 HEADER(1)=ITOTBK
0087 HEADER(2)=IREM
0088 HEADER(3)=ISIZE
C C NOW OPEN FILE
C C---------------------------FILE LOOP
C 0089 DO 300 I=1,ITOTBK
C 0090 IF (I.EQ.1) GOTO 240
C 0092 CALL REQTS(NUMBER,IERROR)
C 0093 IF (IERROR.EQ.1) GOTO 2100
0095 IF (IERROR.EQ.2) GOTO 2200
0097 IF (IERROR.GE.3) GOTO 2300
C 0099 CONTINUE
C C STARTS RECEIVING PROGRAM HERE
C C 0100 CALL RECEVE(IERROR,ILEN,ICSUM,DATA)
C 0101 IF (IERROR.EQ.1) GOTO 3100
0103 IF (IERROR.EQ.2) GOTO 3200
0105 IF (IERROR.GE.3) GOTO 3300
C 0107 IYPE=IDATA(3)
0108 IF (ITYPE.NE.67) GOTO 1100
0110 IBLOCK=IDATA(4)
0111 IF (IBLOCK.NE.IBLKCT) GOTO 4100
C C PUT ARRAY ON DISK.
CALL DISKSV(FILNAM, HEADER, IDATA, LERROR)

IF (LERROR.EQ.'TRUE') GOTO 910

IBLKCT = IBLKCT + 1

CONTINUE

GOTO 499

WRITE (WT, 402)

FORMAT(//' ONLY PROGRAM NUMBERS 1-9 ACCEPTABLE.'

GOTO 10

WRITE (WT, 500) IPNUM

FORMAT(//' PROGRAM NO. 'I4,' SUCCESSFULLY RECEIVED AND FILED',//
1                     ' PRESS 'RETURN' TO GOT MAIN MENU. > ',$)

READ (RT, 904) INPUT

RETURN
C--------------------------------------------------------------------------
C
C           ALL OF THE VARIOUS ERROR MESSAGES FOLLOW:
C
C 0127 900  CONTINUE
C 0128 902  WRITE (WT,902)
C 0129      FORMAT(//,' FATAL ERROR! FILE WILL BE DELETED.'),1
C 0130      READ (RT,904) INPUT
C 0131 904  FORMAT(I4)
C 0132  RETURN
C 0133 910  WRITE (WT,912)
C 0134 912  FORMAT(//,' ERROR RECEIVED FROM SUBROUTINE DISKSV')
C 0135      GOTO 900
C 0136 1000  WRITE (WT,1002)
C 0137 1002  FORMAT(//,' ILLEGAL ACKNOWLEDGE CODE RECEIVED')
C 0138      GOTO 900
C 0139 1100  WRITE (WT,1102)
C 0140 1102  FORMAT(//,' WRONG MESSAGE TYPE RECEIVED')
C 0141      GOTO 900
C 0142 1200  WRITE (WT,1202)
C 0143 1202  FORMAT(//,' WRONG PROGRAM NUMBER RETURNED BY ROBOT')
C 0144      GOTO 900
C 0145 1300  WRITE (WT,1302)
C 0146 1302  FORMAT(//,' ROBOT NOT READY TO RECEIVE',1
C 0147      READ (RT,904) INPUT
C 0148      GOTO 160
C 0149 2100  WRITE (WT,2102)
C 0150 2102  FORMAT(//,' TIME OUT ERROR WHILE WAITING FOR',1
C 0151      GOTO 900
C 0152 2200  WRITE (WT,2202)
C 0153 2202  FORMAT(//,' LRC ERROR MESSAGE RETURNED AFTER TRANSMISSION')
C 0154      GOTO 900
C 0155 2300  WRITE (WT,2302) IERROR
C 0156 2302  FORMAT(//,' ILLEGAL ACKNOWLEDGE CODE',I4,' RECEIVED',1
C 0157      GOTO 900
C 0158 3100  WRITE (WT,3102)
C 0159 3102  FORMAT(//,' FIVE SECOND TIMEOUT OCCURED DURING RECEIVE')
C 0160 3200  WRITE (WT,3202)
C 0161 3202  FORMAT(//,' CHECKSUM ERROR CALCULATION PRODUCED ERROR',1
95
FORTRAN IV

0162      GOTO 900

C
0163  3300 WRITE (WT, 3302) IERROR
0164  3302 FORMAT(//' ILLEGAL ERROR CODE', I4, ' RETURNED FROM',
              1 ' RECEIVE SUBROUTINE."
0165      GOTO 900

C
0166  4100 WRITE (WT, 4102) IBLOCK, IBLKCT
0167  4102 FORMAT(//' BLOCK COUNT TRANSMITTED ', I4, ',
              1 ' DOES NOT MATCH EXPECTED BLOCK ', I4, '"
0168      GOTO 900

C
0169      END
FORTRAN IV  
Storage Map for Program Unit SAVE

Local Variables, .PSECT $DATA, Size = 001114 ( 294, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>R*8</td>
<td>001052</td>
<td>FILNO</td>
<td>I*2</td>
<td>001046</td>
<td>I</td>
<td>I*2</td>
<td>001062</td>
</tr>
<tr>
<td>IACK</td>
<td>I*2</td>
<td>001076</td>
<td>IBLKCT</td>
<td>I*2</td>
<td>001104</td>
<td>IBLOCK</td>
<td>I*2</td>
<td>001112</td>
</tr>
<tr>
<td>ICHAN</td>
<td>I*2</td>
<td>001064</td>
<td>ICSUM</td>
<td>I*2</td>
<td>001072</td>
<td>IERROR</td>
<td>I*2</td>
<td>001066</td>
</tr>
<tr>
<td>ILEN</td>
<td>I*2</td>
<td>001070</td>
<td>INPUT</td>
<td>L*1</td>
<td>001051</td>
<td>IPNUM</td>
<td>I*2</td>
<td>001100</td>
</tr>
<tr>
<td>IREM</td>
<td>I*2</td>
<td>001110</td>
<td>ISIZE</td>
<td>I*2</td>
<td>001102</td>
<td>ITOTBK</td>
<td>I*2</td>
<td>001106</td>
</tr>
<tr>
<td>ITYPE</td>
<td>I*2</td>
<td>001074</td>
<td>LERROR</td>
<td>L*1</td>
<td>001050</td>
<td>NUMBER</td>
<td>I*2</td>
<td>001040</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>001042</td>
<td>WT</td>
<td>I*2</td>
<td>001044</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILNAM</td>
<td>L*1</td>
<td>$DATA</td>
<td>001010</td>
<td>000017</td>
<td>( 8.) (15)</td>
</tr>
<tr>
<td>HEADER</td>
<td>I*2</td>
<td>$DATA</td>
<td>001002</td>
<td>000006</td>
<td>( 3.) (3)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000000</td>
<td>001002</td>
<td>(257.) (257)</td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>$DATA</td>
<td>001027</td>
<td>000007</td>
<td>( 4.) (7)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKSV</td>
<td>R*4</td>
<td>ICLOSE</td>
<td>I*2</td>
<td>IFETCH</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
</tr>
<tr>
<td>INSERT</td>
<td>I*2</td>
<td>IRAD50</td>
<td>I*2</td>
<td>LOOKUP</td>
<td>I*2</td>
<td>RECEIVE</td>
<td>R*4</td>
<td>REDTS</td>
<td>R*4</td>
</tr>
<tr>
<td>SCOPY</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ C - 2 \]

97
SUBROUTINE REQTS(NUMBER, ERROR)

INTEGER*2 IDATA(257), NUMBER, ERROR

SEND REQUEST TO SAVE PROGRAM [NUMBER] TO ROBOT

LENGTH OF MESSAGE

IDATA(1) = 2

SEQUENCE NO. SET TO ZERO

IDATA(2) = 0

SET TYPE CODE

IDATA(3) = 193

SEND PROGRAM NUMBER

IDATA(4) = NUMBER

ILEN = 4

CALL SEND(IERROR, ILEN, IDATA)

ERROR = IERROR

RETURN

END
FORTRAN IV  

Storage Map for Program Unit REQTS

Local Variables, .PSECT $DATA, Size = 001012 ( 261. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>I*2</td>
<td>000002</td>
<td>IERROR</td>
<td>I*2</td>
<td>001010</td>
<td>ILEN</td>
<td>I*2</td>
<td>001006</td>
</tr>
<tr>
<td>NUMBER</td>
<td>I*2</td>
<td>000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000004</td>
<td>001002</td>
<td>(257.) (257)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEND</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE LOAD

PURPOSE:

THIS ROUTINE ALLOWS THE USER TO SELECT A PARTICULAR PROGRAM TO BE LOADED FROM THE MINC DISKETTE TO THE ROBOT. THEN THE PROGRAM IS TRANSMITTED TO THE ROBOT WITH APPROPRIATE HANDSHAKING.

INPUT:

PROGRAM REQUESTS THE NAME OF THE DISKETTE FILE CONTAINING THE PROGRAM TO BE LOADED. PROGRAM ASSUMES '.CYR' FILENAME EXTENSION. THEN IT REQUESTS THE NUMBER BETWEEN 1 AND 9 BY WHICH THE PROGRAM WILL BE REFERENCED IN THE ROBOT.

PROCESSING:

HANDSHAKING:

Computer transmits Type Code 194 - REQUEST TO LOAD PROGRAM FROM COMPUTER.
The computer waits for Type Code 65 - LOAD PROGRAM FROM COMPUTER ACKNOWLEDGE.
When computer is ready to transmit the program it transmits a block of code preceded by Type Code 195.
Then, the robot accepts and files a sequence of blocks using Type Code 65, LOAD PROGRAM FROM COMPUTER ACKNOWLEDGE, to indicate when it is ready for each block.

OUTPUT: LOADS PROGRAM FROM MINC COMPUTER TO ROBOT MEMORY.

CALLED BY: CYR02

CALLS TO: DISKRD, SEND, RECEIVE, INSERT, SCOPY

SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULES.

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
7-13-84 FRED R. SIAS, JR. (803)-656-3375/DESIGN AND PROGRAM
6-03-85 FRED R. SIAS, JR. (803)-656-3375/REV. NARRATION & PROG.
7-23-85 J. KEITH MCELVEEN (803)656-3375/REVISED PROGRAM
9-6-85 FRS ADDITIONAL DEBUGGING
INTEGER*2 NUMBER, IDATA(257), HEADER(3), RT, WT, LSIZE, HSIZE
INTEGER*2 ILEN, IBLOCK, IERROR
LOGICAL*1 LERROR
BYTE INPUT, FILNAM(15), NAMFIL(7)

CALL SCOPY('DY1', FILNAM)
DATA FILNAM(13) = '/00/
DATA NAMFIL(7) = '/00/
WT = 7
RT = 5
CONTINUE

WRITE (WT, 100)
FORMAT(///, 'Input FILE NAME of program to be loaded, ', 1, 'To EXIT, press RETURN.', '')
READ (RT, 105) (NAMFIL(I), I = 1, 6)
FORMAT(6A1)
IF (NAMFIL(1).EQ.' ') RETURN
WRITE (MT, 110)
FORMAT///, 'Input NUMBER program will have in ROBOT, ', '')
READ (RT, 115) NUMBER
FORMAT(I2)
IF (NUMBER.LT.1.OR.NUMBER.GT.9) GOTO 400
CALL INSERT(NAMFIL, FILNAM, 4, 6)
CONTINUE

FIRST RETRIEVE FILE AND GET HEADER SO KNOW PROGRAM SIZE
WRITE (WT, 118) (FILNAM(I), I = 1, 14)
FORMAT///, 'NOW OPENING ', 14A1//
-------------------PASS A ZERO TO INDICATE ONLY WANT HEADER
HEADER(3) = 0
LERROR = 'FALSE'
CALL DISKRD(FILNAM, HEADER, IDATA, IBLOCK, LERROR)
IF (LERROR.EQ.'TRUE') GOTO 910

NOW CALCULATE HSIZE AND LSIZE
ISIZE = HEADER(3)
ITOTBK = HEADER(1)
IREM = HEADER(2)
HSIZE = ISIZE/256
LSIZE = ISIZE-HSIZE*256
NOW TRANSMIT A "REQUEST TO LOAD" TO ROBOT

IDATA(1)=4
IDATA(2)=0
IDATA(3)=194
IDATA(4)=NUMBER
IDATA(5)=LSIZE
IDATA(6)=HSIZE
ILEN=6

WRITE (WT,200)
200 FORMAT(/'TRANSMIT REQUEST TO LOAD CODE 194,/) CALL SEND(IERROR,ILEN,IDATA)

IF (IERROR.EQ.1) GOTO 2100
IF (IERROR.EQ.2) GOTO 2200
IF (IERROR.GE.3) GOTO 2300

DO 300 IBLKCT=1,ITOTBK

NOW WAIT FOR LOAD ACKNOWLEDGE - TYPE CODE 65
WRITE (WT,210)
210 FORMAT(/'RECEIVE LOAD ACKNOWLEDGE TYPE CODE 65,/) CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)

ITYPE=IDATA(3)
IACK=IDATA(4)
IPNUM=IDATA(5)

WRONG MESSAGE TYPE RECEIVED
IF (ITYPE.NE.65) GOTO 1100
IF (IPNUM.NE.NUMBER) GOTO 1200
NOT READY TO LOAD CODE = 0
IF (IACK.EQ.0) GOTO 1300
IF (IACK.GE.2) GOTO 1000
FOUND IACK=1
REACH THIS POINT IF READY TO START SENDING PROGRAM TO ROBOT
C SENDS BLOCK NUMBER AND 252 BYTES OF DATA IN EACH BLOCK
C
C STARTS SENDING REST OF PROGRAM HERE
C CALLS DISKRD WITH IBLOCK SET EQUAL TO EXPECTED BLOCK
C AND CHECKS EACH READ BLOCK AGAINST BLOCK COUNT AFTER
C RETURNING FROM SUBROUTINE DISKRD
C
0073 IBLOCK=IBLKCT-1
0074 CALL DISKRD(FILNAME,HEADER,IDATA,IBLOCK,IERRO)
C
0075 IF (IBLOCK .NE. IDATA(4)) GOTO 4100
C
0077 IDATA(3)=195
0078 ILEN=IDATA(1)+2
C
0079 WRITE (WT,240) IBLOCK
0080 240 FORMAT(′ ′, ′,' SENDING BLOCK ′,I4,   OF PROGRAM,′)
C
0081 CALL SEND(IERROR,ILEN,IDATA)
C
0082 IF (IERROR.EQ.1) GOTO 2100
0084 IF (IERROR.EQ.2) GOTO 2200
0086 IF (IERROR.GE.3) GOTO 2300
C
0088 300 CONTINUE
C------------------------------------------END LOOP
C------------------------------------------
C
C NOW WAIT FOR LOAD ACKNOWLEDGE - TYPE CODE 65
C
C WRITE (WT,210)
C210 FORMAT(′ ′, ′,' RECEIVE LOAD ACKNOWLEDGE TYPE CODE 65,′,/)}
C
C CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)
C
C IF (IERROR.EQ.1) GOTO 3100
C IF (IERROR.EQ.2) GOTO 3200
C IF (IERROR.GE.3) GOTO 3300
C
C ITYPE=IDATA(3)
C IACK=IDATA(4)
C IPNUM=IDATA(5)
C
C WRONG MESSAGE TYPE RECEIVED
C
C IF (ITYPE .NE. 65) GOTO 1100
C IF (IPNUM .NE. NUMBER) GOTO 1200
C NOT READY TO LOAD CODE = 0
C
C IF (IACK.EQ.0) GOTO 1300
IF (IACK .GE. 2) GOTO 1000

! FOUND IACK = 1

WRITE (WT, 310) (NAMFIL(I), I=1,6), NUMBER

READ (RT, 904) INPUT

RETURN

WRITE (WT, 402)

FORMAT ("ONLY PROGRAM NUMBERS 1-9 ACCEPTABLE.")

GOTO 10
ALL OF THE VARIOUS ERROR MESSAGES FOLLOW:

```
0096 900 CONTINUE
0097  WRITE (WT,902)
0098 902 FORMAT(' FATAL ERROR!',$/
1       ' PRESS RETURN TO GOTO MAIN MENU> ',$)
0099  READ (RT,904) INPUT
0100 904 FORMAT(I4)
0101  RETURN
0102 910 CONTINUE
0103  WRITE (WT,912)
0104 912 FORMAT(' ERROR RECEIVED FROM SUBROUTINE DISKRD')
0105  GOTO 10

0106 1000 WRITE (WT,1002)
0107 1002 FORMAT(' ILLEGAL ACKNOWLEDGE CODE RECEIVED,' )
0108  GOTO 900

0109 1100 WRITE (WT,1102)
0110 1102 FORMAT(' WRONG MESSAGE TYPE RECEIVED!')
0111  GOTO 900

0112 1200 WRITE (WT,1202)
0113 1202 FORMAT(' WRONG PROGRAM NUMBER RETURNED BY ROBOT!')
0114  GOTO 900

0115 1300 WRITE (WT,1302)
0116 1302 FORMAT('ROBOT NOT READY TO LOAD PROGRAM!',$/
1       ' TYPE "RETURN" TO TRY AGAIN> ',$)
0117  READ (RT,904) INPUT
0118  GOTO 120

0119 2100 WRITE (WT,2102)
0120 2102 FORMAT(' TIME OUT ERROR WHILE WAITING FOR SEND!',$/
1       ' ACKNOWLEDGE FROM ROBOT. ')
0121  GOTO 900

0122 2200 WRITE (WT,2202)
0123 2202 FORMAT(' LRC ERROR MESSAGE RETURNED AFTER TRANSMISSION. ')
0124  GOTO 900

0125 2300 WRITE (WT,2302) IERROR
0126 2302 FORMAT(' ILLEGAL ACKNOWLEDGE CODE',I4,' RECEIVED',$/
1       ' AFTER TRANSMISSION TO ROBOT. ')
0127 3100 WRITE (WT,3102)
0128 3102 FORMAT(' FIVE SECOND TIMEOUT OCCURED DURING RECEIVE. ')
0129  GOTO 900

0130 3200 WRITE (WT,3202)
0131 3202 FORMAT(' CHECKSUM ERROR CALCULATION PRODUCED ERROR',$)
```
1 ' DURING RECEIVE."
0132 GOTO 900

C
0133 3300 WRITE (WT,3302) IERROR
0134 3302 FORMAT(//,' ILLEGAL ERROR CODE',I4,' RETURNED FROM',
1 ' RECEIVE SUBROUTINE."
0135 GOTO 900

C
0136 4100 WRITE (WT,4102)
0137 4102 FORMAT(//,' BLOCK NUMBER AND BLOCK COUNT DO NOT MATCH."
0138 GOTO 900

C
0139 END
FORTRAN IV  Storage Map for Program Unit LOAD

Local Variables, .PSECT $DATA, Size = 001110 (292, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSIZE</td>
<td>I*2</td>
<td>001054</td>
<td>I</td>
<td>I*2</td>
<td>001066</td>
<td>IACK</td>
<td>I*2</td>
<td>001104</td>
</tr>
<tr>
<td>IBLKCT</td>
<td>I*2</td>
<td>001076</td>
<td>IBLOCK</td>
<td>I*2</td>
<td>001060</td>
<td>ICSUM</td>
<td>I*2</td>
<td>001100</td>
</tr>
<tr>
<td>IERROR</td>
<td>I*2</td>
<td>001062</td>
<td>ILEN</td>
<td>I*2</td>
<td>001056</td>
<td>INPUT</td>
<td>L*1</td>
<td>001065</td>
</tr>
<tr>
<td>IPNUM</td>
<td>I*2</td>
<td>001106</td>
<td>IREM</td>
<td>I*2</td>
<td>001074</td>
<td>ISIZE</td>
<td>I*2</td>
<td>001070</td>
</tr>
<tr>
<td>ITOTBK</td>
<td>I*2</td>
<td>001072</td>
<td>ITYPE</td>
<td>I*2</td>
<td>001102</td>
<td>LERROR</td>
<td>L*1</td>
<td>001064</td>
</tr>
<tr>
<td>LSIZE</td>
<td>I*2</td>
<td>001052</td>
<td>NUMBER</td>
<td>I*2</td>
<td>001044</td>
<td>RT</td>
<td>I*2</td>
<td>001046</td>
</tr>
<tr>
<td>WT</td>
<td>I*2</td>
<td>001050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILNAM</td>
<td>L*1</td>
<td>$DATA</td>
<td>001010</td>
<td>000017 (8,) (15)</td>
<td></td>
</tr>
<tr>
<td>HEADER</td>
<td>I*2</td>
<td>$DATA</td>
<td>001002</td>
<td>000006 (3,) (3)</td>
<td></td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000000</td>
<td>001002 (257,) (257)</td>
<td></td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>$DATA</td>
<td>001027</td>
<td>000007 (4,) (7)</td>
<td></td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKRD</td>
<td>R*4</td>
<td>INSERT</td>
<td>I*2</td>
<td>RECEIVE</td>
<td>R*4</td>
<td>SCOPY</td>
<td>R*4</td>
<td>SEND</td>
<td>R*4</td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE DISKSV(NAMFIL, HEADER, IDATA, LERROR)

PURPOSE: TO SAVE SUCCESSIVE BLOCKS OF ROBOT PROGRAM ON DISKETTE

INPUT: IS PASSED HEADER AND SUCCESSIVE BLOCKS OF DATA

PROCESSING: KEEPS TRACK OF NUMBER OF BLOCKS OF DATA

OUTPUT: WRITES HEADER ON DISKETTE ONE TIME FOLLOWED BY THE SUCCESSIVE BLOCKS OF PROGRAM DATA.

CALLED BY: SAVE

CALLS TO: NONE

SPECIAL INTERFACE REQUIREMENTS: NONE

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS

6-20-85 FRED R. SIAS, JR. (803)656-3375/ NARRATION

7-02-85 FRS NULLED ARRAY AFTER DATA

7-23-85 J. KEITH MCELVEEN (803)656-3375/REVISED PROGRAM

C******************************************************************************
LOGICAL*LERROR
BYTE CR,NAMFIL(15),ARRAY(5000),BPDATA(520)
INTEGER*2 IARRAY(2500),IPDATA(260),ITOTBK
INTEGER*2 HEADER(3),IData(260),LASTBK,IREM,ISIZE,IBLOCK,RT,WT
REAL*8 FILE1,FILE

DATA FILE1 /12RDY1PROGRMCYR/
COMMON /INOUT/ ARRAY,BPDATA
EQUIVALENCE (ARRAY,IAARRAY)
EQUIVALENCE (BPDATA,IPDATA)
DATA CR /'015/
WT=7
RT=5
IERROR=IRAD50(12,NAMFIL,FILE)

TAKE BLOCK OF DATA AND PUT IN LOCAL ARRAY
DO 10, I=1,256
IPDATA(I)=IData(I)
CONTINUE
NOW DECIDE WHAT TO DO
IBLOCK=DATA(4)
HERE TO GET HEADER AND FIRST BLOCK OF DATA
ITOTBK=HEADER(1)
IREM =HEADER(2)
ISIZE =HEADER(3)
LASTBK=ITOTBK-1
IF (IBLOCK.EQ.0) INDEX=1
FIRST BLOCK SAME AS LAST
IF(IBLOCK.EQ.LASTBK) GOTO 300
PUT THIS BLOCK IN BIG ARRAY
CONTINUE
USE THIS FOR FULL BLOCKS OF DATA
DO 299, I=5,256
IF (IPDATA(I).EQ.'012) ARRAY(INDEX)=CR
IF (IPDATA(I).EQ.'012) INDEX=INDEX+1
C
0033  ARRAY(INDEX)=BPDATA(I*2-1)
0034         INDEX=INDEX+1
C
0035 299 CONTINUE
C
RETURN IF NOT LAST BLOCK OF DATA
C
0036  IF (IBLOCK.NE.LASTBK) RETURN
C
0038       GOTO 400
C
0039 300 CONTINUE
C
USE THIS FOR LAST BLOCK OF DATA
C
0040       DO 399, I=5,IREM+4
C
0041       IF (IPDATA(I).EQ.'012)  ARRAY(INDEX)=CR
0043       IF (IPDATA(I).EQ.'012)  INDEX=INDEX+1
C
0045  ARRAY(INDEX)=BPDATA(I*2-1)
0046         INDEX=INDEX+1
C
0047 399 CONTINUE
C
0048 400 CONTINUE
C
ADD NULLS TO END OF BLOCK OF ARRAY
C
0049       ILAST=INDEX-1
0050       IADD=MOD(ILAST,512)
0051       IEND=INDEX+IADD
0052       DO 450, I=INDEX,IEND
0053             ARRAY(I)=0
C
C
0054 450 CONTINUE
CONTINUE
ICHAN=IGETC()
IF(ICHAN.LT.0) STOP 'NO CHANNEL AVAILABLE'
CREATE OUTPUT FILE
IBLK=0
IERROR=IENTER(ICHAN, FILE, 0)
IF (IERROR.EQ.-2) GOTO 1040
IF (IERROR.LT.0) STOP 'ENTER FAILURE'
IEND=IEND/2
NWORDS=IWRITW(IEND, IARRAY, IBLK, ICHAN)
IF (NWORDS.LT.0) STOP 'ERROR WRITING TO DISKETTE'
IBLK=IBLK+1
IF (IBLK.NE.LASTBK) GOTO 480
CALL ICLOSE(ICHAN, IERROR)
IF (IERROR.LT.0) STOP 'ERROR CLOSING CHANNEL'
CALL IFREEC(ICHAN)
NORMAL EXIT HERE
LERROR='FALSE'
RETURN
ERROR MESSAGES AND ERROR RETURN
WRITE (WT, 1002) NAMFIL
FORMAT(//', ERROR OPENING FILE ',14A1)
GOTO 1100
WRITE (WT, 1022) NAMFIL
FORMAT(//', ERROR WRITING DATA TO FILE ',14A1)
GOTO 1100
WRITE (WT, 1032) NAMFIL
FORMAT(//', ERROR CLOSING FILE ',14A1)
GOTO 1100
CALL ICLOSE(ICHAN)
0090 CALL IFREEC(ICHAN)
0091 WRITE (UTIO42)
0092 1042 FORMAT(' ', '/ ', 'ERROR! DISKETTE FULL!')
C
C ERROR RETURN
C
0093 1100 CONTINUE
C
0094 LERROR='TRUE'
0095 RETURN
0096 END
FORTRAN IV

Storage Map for Program Unit DISKSV

Local Variables, .PSECT $DATA, Size = 000112 ( 37. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>I*2</td>
<td>000020</td>
<td>FILE</td>
<td>R*8</td>
<td>000052</td>
<td>FILE1</td>
<td>R*8</td>
<td>000010</td>
</tr>
<tr>
<td>I</td>
<td>I*2</td>
<td>000064</td>
<td>IADD</td>
<td>I*2</td>
<td>000072</td>
<td>IBLK</td>
<td>I*2</td>
<td>000100</td>
</tr>
<tr>
<td>IBLOCK</td>
<td>I*2</td>
<td>000044</td>
<td>ICHAN</td>
<td>I*2</td>
<td>000076</td>
<td>IEND</td>
<td>I*2</td>
<td>000074</td>
</tr>
<tr>
<td>IERROR</td>
<td>I*2</td>
<td>000062</td>
<td>ILAST</td>
<td>I*2</td>
<td>000070</td>
<td>INDEX</td>
<td>I*2</td>
<td>000066</td>
</tr>
<tr>
<td>IREM</td>
<td>I*2</td>
<td>000040</td>
<td>ISIZE</td>
<td>I*2</td>
<td>000042</td>
<td>ITOTBK</td>
<td>I*2</td>
<td>000034</td>
</tr>
<tr>
<td>LASTBK</td>
<td>I*2</td>
<td>000036</td>
<td>LERROR</td>
<td>L*1</td>
<td>@000006</td>
<td>NWARGS</td>
<td>I*2</td>
<td>000102</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>000046</td>
<td>WT</td>
<td>I*2</td>
<td>000050</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /INOUT /, Size = 012620 ( 2760. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>000000</td>
<td>EAV</td>
<td>BPDATA</td>
<td>L*1</td>
<td>011610</td>
<td>EAV</td>
<td>IARRAY</td>
</tr>
<tr>
<td>IPDATA</td>
<td>I*2</td>
<td>011610</td>
<td>EAV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>INOUT</td>
<td>000000</td>
<td>011610 (2500)</td>
<td>(5000)</td>
</tr>
<tr>
<td>BPDATA</td>
<td>L*1</td>
<td>INOUT</td>
<td>011610</td>
<td>001010 (260)</td>
<td>(520)</td>
</tr>
<tr>
<td>HEADER</td>
<td>I*2</td>
<td>@ $DATA</td>
<td>000002</td>
<td>000006 (3)</td>
<td>(3)</td>
</tr>
<tr>
<td>IARRAY</td>
<td>I*2</td>
<td>INOUT</td>
<td>000000</td>
<td>011610 (2500)</td>
<td>(2500)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>@ $DATA</td>
<td>000004</td>
<td>001010 (260)</td>
<td>(260)</td>
</tr>
<tr>
<td>IPDATA</td>
<td>I*2</td>
<td>INOUT</td>
<td>011610</td>
<td>001010 (260)</td>
<td>(260)</td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>@ $DATA</td>
<td>000000</td>
<td>000017 (8)</td>
<td>(15)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>I*2</td>
<td>IENTER</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
<td>IRAD50</td>
<td>I*2</td>
</tr>
<tr>
<td>IWRTW</td>
<td>I*2</td>
<td>MOD</td>
<td>I*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE DISKRD(NAMFIL, HEADER, IDATA, IBLOCK, LERROR)

PURPOSE: TO READ SUCCESSIVE BLOCKS OF PROGRAM FROM DISKETTE

INPUT: READS HEADER AND SUCCESSIVE BLOCKS OF DATA FROM DISKETTE

PROCESSING: KEEPS TRACK OF NUMBER OF BLOCKS OF DATA

OUTPUT: PASSES PROGRAM BACK TO CALLING ROUTINE IN ARRAY IDATA

CALLED BY: LOAD

CALLS TO: NONE

SPECIAL INTERFACE REQUIREMENTS: NONE

REVISION HISTORY

DATE PROGRAMMER CONTACT/ACTION/REMARKS
6-24-85 FRED R. SIAS, JR. (803)656-3375 NARRATION/PROGRAM
7-05-85 FRS FINISHED DEBUGGING
7-23-85 J. KEITH MCELVEEN (803)656-3375/REVISED PROGRAM
9-6-85 FRS FIXED ARRAY INDEXES/ LOST DATA
LOGICAL*1 LERROR
BYTE NAMFIL(15), ARRAY(5000), BPDATA(520), IBUFF(512)
INTEGER*2 I ARRAY(2500), IPDATA(260)
INTEGER*2 HEADER(3), IDATA(257), LASTBK, IREM, ISIZE, IBLOCK, RT, WT
REAL*8 FILE

COMMON /INOUT/ ARRAY, BPDATA
EQUIVALENCE (ARRAY, I ARRAY)
EQUIVALENCE (BPDATA, IPDATA)

WT=7
RT=5

GO DOWN TO 350 IF NOT CALL FOR HEADER

IF (HEADER(3).NE.0) GOTO 350
CALL IRA50(12, NAMFIL, FILE)
IFLAG=HEADER(3)
OPEN FILE AND READ

ICHAN=IGETC()
IF (ICHAN.LT.0) STOP 'NO CHANNEL AVAILABLE'
IERROR=IFETCH(FILE)
IF (IERROR.NE.0) STOP 'BAD FETCH'
IERROR=LOOKUP(ICHAN, FILE)
IF (IERROR.EQ.-2) GOTO 1040
IF (IERROR.LT.0) STOP 'BAD LOOKUP'
IBLOCK=0
INDEX=1

CONTINUE

CALL IRCLOSE(ICHAN)
CALL IFREEC(ICHAN)
DELETE CR'S IN ARRAY

J=1
DO 190, I=1,5000
IF (ARRAY(I), EQ,'015) GOTO 190
ARRAY(J)=ARRAY(I)
IF (ARRAY(I), EQ,'0) GOTO 200
J=J+1
CONTINUE
CONTINUE
C
C
0054 ISIZE=J-1
0055 ITOTBK=ISIZE/252
0056 IREM=ISIZE-ITOTBK*252
0057 IF (IREM.NE.0) ITOTBK=ITOTBK+1
0059 LASTBK=ITOTBK-1
C
---DIAGNOSTICS---
C
C WRITE (WT,250) ISIZE,IREM,ITOTBK
C250 FORMAT(‘ ISIZE= ’,I6,’ IREM= ’,I6,’ ITOTBK= ’,I6)
C
C WRITE (WT,260) (ARRAY(I),I+1,ISIZE)
C260 FORMAT(‘ ’,6C ’,14))
C
0060 HEADER(1)=ITOTBK
0061 HEADER(2)=IREM
0062 HEADER(3)=ISIZE
C
C RETURN WITH JUST HEADER INFORMATION
C
C
0063 IF (IFLAG.EQ.0) INDEX=1
0065 IF (IFLAG.EQ.0) RETURN
C
C HERE IF MOVING BLOCKS OF DATA
C
0067 350 CONTINUE
C
0068 IF (IBLOCK.EQ.LASTBK) GOTO 400
C
C THIS FOOLISHNESS NEEDED SINCE CAN'T EQUIVALENCE PARAMETER
C
0070 DO 390, I=5,256
0071 BPDATA(I*2-1)=ARRAY(INDEX)
0072 IDATA(I)=IPDATA(I)
0073 INDEX=INDEX+1
0074 K=I
0075 390 CONTINUE
0076 GOTO 500
C
0077 400 CONTINUE
0078 DO 490, I=5,IREM+4
0079 BPDATA(I*2-1)=ARRAY(INDEX)
0080 IDATA(I)=IPDATA(I)
0081 INDEX=INDEX+1
0082 K=I
0083 490 CONTINUE
C
0084 500 CONTINUE
C
C NORMAL EXIT HERE
C
0085 IDATA(4)=IBLOCK
0086 IDATA(1)=K-2
C
0087   LERROR='FALSE'
0088   RETURN
C
C ERROR MESSAGES AND ERROR RETURN
C
0089 1000 CONTINUE
C WRITE (WT,1002) NAMFIL
0090 1002 FORMAT(’/,’ ERROR OPENING FILE ’,14A1,’/,’ DOES IT EXIST?’)
0091 GOTO 1100
C
0092 1010 CONTINUE
0093 WRITE (WT,1012) NAMFIL
0094 1012 FORMAT(’/,’ ERROR READING HEADER FROM FILE ’,14A1)
0095 GOTO 1100
C
0096 1020 CONTINUE
0097 WRITE (WT,1022) NAMFIL
0098 1022 FORMAT(’/,’ ERROR READING DATA FROM FILE ’,14A1)
0099 GOTO 1100
C
0100 1030 CONTINUE
0101 WRITE (WT,1032) NAMFIL
0102 1032 FORMAT(’/,’ ERROR CLOSING FILE ’,14A1)
0103 GOTO 1100
C
0104 1040 CONTINUE
0105 CALL ICLOSE(ICHAN)
0106 CALL IFREEC(ICHAN)
0107 WRITE (WT,1042)
0108 1042 FORMAT(’/,’ FILE NOT FOUND. TRY AGAIN.’)
C
0109 1100 CONTINUE
C
0110 LERROR=’TRUE’
0111 RETURN
0112 END
## FORTRAN IV

**Storage Map for Program Unit DISKRD**

### Local Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>R*8</td>
<td>001040</td>
<td>I</td>
<td>I*2</td>
<td>001060</td>
<td>IBLOCK</td>
<td>I*2</td>
<td>@ 000006</td>
</tr>
<tr>
<td>ICHAN</td>
<td>I*2</td>
<td>001052</td>
<td>IERROR</td>
<td>I*2</td>
<td>001054</td>
<td>IFLAG</td>
<td>I*2</td>
<td>001050</td>
</tr>
<tr>
<td>INDEX</td>
<td>I*2</td>
<td>001056</td>
<td>IREM</td>
<td>I*2</td>
<td>001030</td>
<td>ISIZE</td>
<td>I*2</td>
<td>001032</td>
</tr>
<tr>
<td>ITOTBK</td>
<td>I*2</td>
<td>001064</td>
<td>J</td>
<td>I*2</td>
<td>001062</td>
<td>K</td>
<td>I*2</td>
<td>001066</td>
</tr>
<tr>
<td>LASTBK</td>
<td>I*2</td>
<td>001026</td>
<td>LERROR</td>
<td>L*1  @ 000010</td>
<td>RT</td>
<td>I*2</td>
<td>001034</td>
<td></td>
</tr>
<tr>
<td>WT</td>
<td>I*2</td>
<td>001036</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### COMMON Block

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>000000</td>
<td>BPDATA</td>
<td>L*1</td>
<td>011610</td>
<td>Eav</td>
<td>IARRAY</td>
<td>I*2</td>
</tr>
<tr>
<td>IPDATA</td>
<td>I*2</td>
<td>011610</td>
<td>Eav</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Local and COMMON Arrays

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>INOUT</td>
<td>000000</td>
<td>011610 ( 2500,)</td>
<td>(5000)</td>
</tr>
<tr>
<td>BPDATA</td>
<td>L*1</td>
<td>INOUT</td>
<td>011610</td>
<td>001010 ( 260,)</td>
<td>(520)</td>
</tr>
<tr>
<td>HEADER</td>
<td>I*2</td>
<td>@ $DATA</td>
<td>000002</td>
<td>000006 ( 3,)</td>
<td>(3)</td>
</tr>
<tr>
<td>IARRAY</td>
<td>I*2</td>
<td>INOUT</td>
<td>000000</td>
<td>011610 ( 2500,)</td>
<td>(2500)</td>
</tr>
<tr>
<td>IBUFF</td>
<td>L*1</td>
<td>$DATA</td>
<td>000012</td>
<td>001000 ( 256,)</td>
<td>(512)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>@ $DATA</td>
<td>000004</td>
<td>001002 ( 257,)</td>
<td>(257)</td>
</tr>
<tr>
<td>IPDATA</td>
<td>I*2</td>
<td>INOUT</td>
<td>011610</td>
<td>001010 ( 260,)</td>
<td>(260)</td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>@ $DATA</td>
<td>000000</td>
<td>000017 ( 8,)</td>
<td>(15)</td>
</tr>
</tbody>
</table>

### Subroutines, Functions, Statement and Processor-Defined Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>I*2</td>
<td>IFETCH</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
<td>IRAD50</td>
<td>I*2</td>
</tr>
<tr>
<td>IREADW</td>
<td>I*2</td>
<td>LOOKUP</td>
<td>I*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

120
SUBROUTINE RUN

PURPOSE:

THIS ROUTINE IS USED TO START ANY PROGRAM STORED IN THE ROBOT. THE PROGRAM IS SELECTED BY NUMBER WHICH IS REQUESTED BY THIS PROGRAM AND PASSED TO THE ROBOT AS PART OF THE MESSAGE.

THE LOAD PROGRAM IN THE MINC MAY BE USED TO TRANSFER A PROGRAM FROM THE MINC DISC TO THE ROBOT MEMORY PRIOR TO STARTING THE PROGRAM USING THIS RUN ROUTINE.

INPUT: ACCEPTS PROGRAM NUMBER FROM MINC KEYBOARD.

PROCESSING:

SETS UP MESSAGE AND CONTROLS COMMUNICATION. AFTER TRANSMIT AND ACKNOWLEDGE WILL SIT IN LOOP AND RECEIVE MESSAGES INDEFINITELY. KEYPRESS TO EXIT.

OUTPUT:

SUBROUTINE DISPLAYS APPROPRIATE PROMPT MESSAGE PRIOR TO ACCEPTING A PROGRAM NUMBER FROM KEYBOARD. THE ACTUAL MESSAGE TRANSMITTED IS THE "SET PROGRAM MODE" MESSAGE. MUST BE TRANSMITTED WHILE ROBOT IS IN "NO-ACTIVITY STATE". THE PROGRAM TO BE RUN MUST ALREADY BE STORED IN THE ROBOT CONTROLLER MEMORY.

MESSAGE TYPE CODE = 130

THE KEYSWITCH ON THE ROBOT MUST BE ENABLED.

CALLED BY: CYR02

CALLS TO: SEND, RECEIVE

SPECIAL INTERFACE REQUIREMENTS: MINC DIGITAL I/O MODULES

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
7-10-84 FRED R. SIAS, JR. (803)-656-3375/SYSTEM DESIGN/PROGRAM
10-26-84 FRED R. SIAS, JR. /SIGNIFICANT PROGRAM REVISION
6-4-85 FRED R. SIAS, JR. /REVISED NARRATION
8-10-85 J. KEITH MCELVEEN /REVISED RECEIVE LOOP

*****************************************************************************

121
C 0002 INTEGER*2 WT, RT, IDATA(257), INDATA
C 0003 INTEGER*2 ILEN, IERROR
C 0004 WT = 7
C 0005 RT = 5
C 0006 10 CONTINUE
C 0007 WRITE (WT, 100)
C 0008 100 FORMAT(//, 'Input NUMBER of robot program to RUN.',
C 0009       1 'To EXIT, press RETURN.', $)
C 0010 READ (RT, 110, ERR = 10) INDATA
C 0011 110 FORMAT(I4)
C 0012 IF (INDATA.EQ.0) RETURN
C 0013 IF (INDATA.LT.1 OR INDATA.GT.9) GOTO 400
C 0014 LENGTH OF MESSAGE TO TRANSMIT
C 0015 IDATA(1) = 3
C 0016 SET SEQUENCE NUMBER TO 0
C 0017 IDATA(2) = 0
C 0018 SET TYPE CODE TO 130
C 0019 IDATA(3) = 130
C 0020 SET PROGRAM STATUS TO 'RUN'
C 0021 IDATA(4) = 1
C 0022 TRANSMIT PROGRAM NUMBER
C 0023 IDATA(5) = INDATA
C 0024 ILEN = 5
C 0025 CALL SEND (IERROR, ILEN, IDATA)
C 0026 IF (IERROR.EQ.1) GOTO 1000
C 0027 IF (IERROR.EQ.2) GOTO 2000
C 0028 IF (IERROR.GT.2) GOTO 3000
C 0029 CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)
C 0030 IF (IERROR.EQ.1) GOTO 1000
C 0031 IF (IERROR.EQ.2) GOTO 2000
C 0032 IF (IERROR.GT.2) GOTO 3000
C 0033 WRITE (WT, 1070) INDATA
C 0034 1070 FORMAT(//, 'PROGRAM NO.', I2, ' HAS BEEN STARTED.',
C 0035       1 'IN ROBOT.', $,
C 0036       2 'System will ignore any messages during program run.', $,
C 0037       3 'AFTER RUN, PRESS ANY KEY TO RETURN TO MENU.', $)
C 0038 DO 7000 I = 1, 2500
C 0039 CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)
0039 IF (IERROR.EQ.1) GOTO 6000
0041 IF (IERROR.EQ.2) GOTO 2000
0043 IF (IERROR.GT.2) GOTO 3000

C 0045 7000 CONTINUE
C 0046 TYPE *,' RECEIVED 2500 MESSAGES, CONTINUING'
C
C WILL EXIT ABOVE LOOP IF TIMEOUT OCCURS OR IF RECEIVES
C 2500 MESSAGES.
C
C 0047 6000 CONTINUE
C 0048 RETURN
C
C 0049 400 WRITE(WT,402)
0050 402 FORMAT(//' ONLY PROGRAM NUMBERS 1-9 ACCEPTABLE,')
0051 GOTO 10
C
0052 1000 CONTINUE
0053 WRITE(WT,1010)
0054 1010 FORMAT(//' FIVE SECOND TIMEOUT OCCURRED DURING RECEIVE,')
0055 GOTO 999
C
0056 2000 CONTINUE
0057 WRITE(WT,2010)
0058 2010 FORMAT(//' CHECKSUM ERROR CALCULATION PRODUCED ERROR,',
1 ' DURING RECEIVE,')
0059 GOTO 999
C
0060 3000 CONTINUE
0061 WRITE(WT,3010) IERROR
0062 3010 FORMAT(//' ILLEGAL ERROR CODE',I4,' RETURNED FROM',
1 ' RECEIVE SUBROUTINE,')
C
0063 999 CONTINUE
0064 WRITE(WT,9999)
0065 9999 FORMAT(//' PRESS 'RETURN' TO GOTO MAIN MENU,'*,*)
0066 READ (RT,9998) INPUT
0067 9998 FORMAT (A4)
C
0068 RETURN
0069 END
FORTRAN IV  Storage Map for Program Unit RUN

Local Variables, .PSECT $DATA, Size = 001022 ( 265, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>001016</td>
<td>ILEN</td>
<td>I*2</td>
<td>001010</td>
<td>RT</td>
<td>I*2</td>
<td>001004</td>
</tr>
<tr>
<td>ILEN</td>
<td>I*2</td>
<td>001010</td>
<td>ICSUM</td>
<td>I*2</td>
<td>001014</td>
<td>IERROR</td>
<td>I*2</td>
<td>001012</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>001004</td>
<td>INDATA</td>
<td>I*2</td>
<td>001006</td>
<td>INPUT</td>
<td>I*2</td>
<td>001020</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section Offset</th>
<th>——-Size——-</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000000</td>
<td>001002</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td>SEND</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE DIRECT

PURPOSE:
ROUTINE ACCEPTS DIR COMMAND FROM TERMINAL, THEN A REQUEST IS TRANSMITTED TO THE ROBOT AND THE COMPUTER RECEIVES AND DISPLAYS THE PROGRAM NUMBER AND THE FIRST TWO LINES OF EACH PROGRAM ON THE MINC CRT.

INPUT: NONE

PROCESSING:

HANDSHAKING:
Computer transmits Type Code 193 - REQUEST TO SAVE PROGRAM TO COMPUTER.
The computer waits for Type Code 66 - SAVE PROGRAM TO COMPUTER ACKNOWLEDGE.

Then computer accepts a sequence of blocks using Type Code 193 to indicate when it is ready for each block.
Each block of data transmitted to the robot has a header containing the Type Code 67 followed by a sequential block number and the actual N/C program data.

OUTPUT: DISPLAYS APPROPRIATE PROMPTS ON MINC CRT AND THEN DISPLAYS THE PROGRAM NUMBER AND THE FIRST TWO LINES OF EACH PROGRAM ON THE MINC CRT.

CALLED BY: CYRO2

CALLS TO: REQTS, RECEIVE

SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULES

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

<table>
<thead>
<tr>
<th>DATE</th>
<th>PROGRAMMER</th>
<th>CONTACT/ACTION/REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-27-85</td>
<td>FRED R. SIAS, JR.</td>
<td>(803)-656-3375/SYSTEM DESIGN/PROGRAM</td>
</tr>
<tr>
<td>6-04-85</td>
<td>FRED R. SIAS, JR.</td>
<td>/REVISED NARRATION</td>
</tr>
<tr>
<td>6-10-85</td>
<td>FRED R. SIAS, JR.</td>
<td>/REVISED LOGIC</td>
</tr>
</tbody>
</table>

**********
C INTEGER*2 NUMBER,RT,WT,FILNO,OUTDV
C BYTE INPUT
C LOGICAL*1 LERROR
C INCLUDE 'COMMON.FOR' ---NOT LEGAL IN THIS FORTRAN
C INTEGER*2 IDATA(258)
C BYTE PROG(1024),BUFFER(80)
C COMMON IDATA,PROG,BUFFER
C
C WT=7
C RT=5
C OUTDV=7
C
C CONTINUE
C
C DO 400, NUMBER=1,9
C
C CONTINUE
C
C NOW TRANSMIT A 'REQUEST TO SAVE' TO ROBOT
C
C CALL REQTS(NUMBER,LERROR)
C
C IF (LERROR.EQ.1) GOTO 2100
C IF (LERROR.EQ.2) GOTO 2200
C IF (LERROR.GE.3) GOTO 2300
C
C NOW WAIT FOR SAVE ACKNOWLEDGE - TYPE CODE 66
C
C CONTINUE
C
C CALL RECEVE(IERROR,ILEN,ICSUM,IDATA)
C
C IF (IERROR.EQ.1) GOTO 3100
C IF (IERROR.EQ.2) GOTO 3200
C IF (IERROR.GE.3) GOTO 3300
C
C ITYPE=IDATA(3)
C IACK=IDATA(4)
C IPNUM=IDATA(5)
C ISIZE=IDATA(6)+256*IDATA(7)
C
C WRONG MESSAGE TYPE RECEIVED
C
C IF (ITYPE.NE.66) GOTO 1100
C
C IF (IPNUM.NE.NUMBER) GOTO 1200
C
C NOT READY TO RECEIVE CODE = 0
C
C IF (IACK.EQ.0) GOTO 1300
C IF (IACK.GE.2) GOTO 1000
URITE(WT,203) NUMBER
FORMAT( ' Program No. ',I2,' ')
IBLKCT=0
RECEIVES BLOCK NUMBER AND 252 BYTES OF DATA IN EACH BLOCK
ITOTBK=ISIZE/252
IREM=ISIZE-ITOTBK*252
IF (IREM.NE.0) ITOTBK=ITOTBK+1
LOOP TO RECEIVE AND PUT IN ARRAY FOLLOWS:
DO 300 I=1,ITOTBK
SKIP REQUEST TO SEND FOR FIRST BLOCK OF DATA
IF (I.EQ.1) GOTO 240
CALL REQTS(NUMBER,IERROR)
IF (IERROR.EQ.1) GOTO 2100
IF (IERROR.EQ.2) GOTO 2200
IF (IERROR.GE.3) GOTO 2300
CONTINUE
STARTS RECEIVING PROGRAM HERE
CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)
IF (IERROR.EQ.1) GOTO 3100
IF (IERROR.EQ.2) GOTO 3200
IF (IERROR.GE.3) GOTO 3300
ITYPE=IDATA(3)
IF (ITYPE.NE.67) GOTO 1100
IBLOCK=IDATA(4)
IF (IBLOCK.NE.IBLKCT) GOTO 4100
DISPLAY ARRAY ON CRT
IF (IBLKCT.NE.0) GOTO 260
ICRS=1
DO 250 K=5,ILEN+2
IF(IDATA(K).EQ."012) WRITE (OUTDV,205)
FORMAT("+",/)
IF(IDATA(K).EQ."012) ICRS=ICRS+1
C
0081 IF(IDATA(K),EQ.0) GOTO 260
0083 IF(ICRS,EQ.2) GOTO 260
C
0085 IF(IDATA(K),NE,'012) WRITE (OUTDV,220) IDATA(K)
0087 220 FORMAT('+',A1,*)
C
0088 250 CONTINUE
C
0089 260 CONTINUE
C
0090 IBLKCT=IBLKCT+1
C END OF FILE TRANSFER LOOP
C
0091 300 CONTINUE
C
0092 400 CONTINUE
C
0093 WRITE (WT,500)
0094 500 FORMAT('PRESS 'RETURN' TO GOTO MAIN MENU,> ','*)
0095 READ (RT,904) INPUT
C
0096 RETURN
ALL OF THE VARIOUS ERROR MESSAGES FOLLOW:

CONTINUE

WRITE (WT,992) FORMAT(//' PRESS RETURN TO GOTO MAIN MENU> '),$)
READ (RT,904) INPUT
FORMAT(I4)
RETURN

WRITE (WT,1002) FORMAT(//' ILLEGAL ACKNOWLEDGE CODE RECEIVED,')
GOTO 900

WRITE (WT,1102) FORMAT(//' WRONG MESSAGE TYPE RECEIVED!')
GOTO 900

WRITE (WT,1202) FORMAT(//' WRONG PROGRAM NUMBER returned by robot!')
GOTO 900

WRITE (WT,1302) FORMAT(//' ROBOT NOT READY TO RECEIVE,'),
   ' TYPE 'RETURN' TO TRY AGAIN> ',$)
READ (RT,904) INPUT
GOTO 120

WRITE (WT,2102) FORMAT(//' TIME OUT ERROR WHILE WAITING FOR' ),
   ' ACKNOWLEDGE FROM ROBOT.')
GOTO 900

WRITE (WT,2202) FORMAT(//' LRC ERROR MESSAGE returned after transmission.')
GOTO 900

WRITE (WT,2302) FORMAT(//' ILLEGAL ACKNOWLEDGE CODE',I4,' RECEIVED',
   ' AFTER TRANSMISSION TO ROBOT.')

WRITE (WT,3102) FORMAT(//' FIVE SECOND TIMEOUT OCCURRED DURING RECEIVE.')
GOTO 900

WRITE (WT,3202) FORMAT(//' CHECKSUM ERROR CALCULATION PRODUCED ERROR',
   ' DURING RECEIVE.')
GOTO 900

WRITE (WT,3302) FORMAT(//' ILLEGAL ERROR CODE',I4,' RETURNED FROM',
   ' RECEIVE SUBROUTINE.')
GOTO 900

WRITE (WT,4102) IBLOCK,IBLKCT
0134 4102 FORMAT(/, ' BLOCK COUNT TRANSMITTED (',I4, '),' ,
   1   ' DOES NOT MATCH EXPECTED BLOCK (',I4, ').'

0135 C GOTO 900

0136 CEND
FORTRAN IV  
Storage Map for Program Unit DIRECT

Local Variables, .PSECT $DATA, Size = 000056 ( 23, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILNO</td>
<td>I*2</td>
<td>000012</td>
<td>I</td>
<td>I*2</td>
<td>000044</td>
<td>IACK</td>
<td>I*2</td>
<td>000030</td>
</tr>
<tr>
<td>IBLKCT</td>
<td>I*2</td>
<td>000036</td>
<td>IBLOCK</td>
<td>I*2</td>
<td>000046</td>
<td>ICMS</td>
<td>I*2</td>
<td>000050</td>
</tr>
<tr>
<td>ICSUM</td>
<td>I*2</td>
<td>000024</td>
<td>IERROR</td>
<td>I*2</td>
<td>000020</td>
<td>ILEN</td>
<td>I*2</td>
<td>000022</td>
</tr>
<tr>
<td>INPUT</td>
<td>L*1</td>
<td>000016</td>
<td>IPNUM</td>
<td>I*2</td>
<td>000032</td>
<td>IREM</td>
<td>I*2</td>
<td>000042</td>
</tr>
<tr>
<td>ISIZE</td>
<td>I*2</td>
<td>000034</td>
<td>ITOTBK</td>
<td>I*2</td>
<td>000040</td>
<td>ITYPE</td>
<td>I*2</td>
<td>000026</td>
</tr>
<tr>
<td>K</td>
<td>I*2</td>
<td>000052</td>
<td>LERROR</td>
<td>L*1</td>
<td>000017</td>
<td>NUMBER</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>OUTDV</td>
<td>I*2</td>
<td>000014</td>
<td>RT</td>
<td>I*2</td>
<td>000006</td>
<td>WT</td>
<td>I*2</td>
<td>000010</td>
</tr>
</tbody>
</table>

COMMON Block /   /, Size = 003124 ( 810, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>000000</td>
<td>PROG</td>
<td>L*1</td>
<td>001004</td>
<td>BUFFER</td>
<td>L*1</td>
<td>003004</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER</td>
<td>L*1</td>
<td>.****</td>
<td>003004</td>
<td>00120 (40)</td>
<td>(80)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>.****</td>
<td>000000</td>
<td>001004 (258)</td>
<td>(258)</td>
</tr>
<tr>
<td>PROG</td>
<td>L*1</td>
<td>.****</td>
<td>000000</td>
<td>002000 (512)</td>
<td>(1024)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td>REQTS</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME:

SUBROUTINE DISPLA

PURPOSE:

ROUTINE ACCEPTS PROGRAM NUMBER TO BE DISPLAYED FROM TERMINAL,
THEN A REQUEST IS TRANSMITTED TO THE ROBOT AND THE COMPUTER
RECEIVES AND DISPLAYS THE PROGRAM RETURNED TO IT.

INPUT: ACCEPTS PROGRAM NUMBER FROM MINC KEYBOARD.

PROCESSING:

HANDSHAKING:

Computer transmits Type Code 193 - REQUEST TO SAVE
PROGRAM TO COMPUTER.
The computer waits for Type Code 66 - SAVE PROGRAM
TO COMPUTER ACKNOWLEDGE.

6-4-85 ACCORDING TO RUSS VIRES ROBOT NO LONGER WAITS
FOR FIRST TYPE CODE 193 - NOT IN CURRENT DOCUMENTATION -
RATHER TWO MESSAGES FOLLOW IN SEQUENCE SEPARATED BY A
BRIEF PAUSE. SECOND MESSAGE IS FIRST BLOCK OF DATA.

Then computer accepts a sequence of blocks
using Type Code 193 to indicate when
it is ready for each block.
Each block of data transmitted to the robot has a header
containing the Type Code 67 followed by
a sequential block number and the actual
N/C program data.

OUTPUT: DISPLAYS APPROPRIATE PROMPTS ON MINC CRT AND THEN
DISPLAYS SELECTED ROBOT PROGRAM ON MINC CRT OR PRINTER.

CALLED BY: CYRO2

CALLS TO: REQTS, RECEIVE

SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULES

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
5-27-85 FRED R. SIAS, JR. (803)-656-3375/SYSTEM DESIGN/PROGRAM
6-04-85 FRED R. SIAS, JR. /REVISED NARRATION
6-10-85 FRED R. SIAS, JR. /REVISED LOGIC

******************************************************************************
C 0002 INTEGER*2 NUMBER, RT, WT, FILNO, OUTDV
C 0003 BYTE INPUT
C 0004 LOGICAL*1 LERROR
C INCLUDE 'COMMON, FOR' --- NOT LEGAL IN THIS FORTRAN
C
C 0005 INTEGER*2 IDATA(258)
C 0006 BYTE PROG(1024), BUFFER(80)
C 0007 COMMON IDATA, PROG, BUFFER
C
C 0008 WT=7
C 0009 RT=5
C 0010 OUTDV=7
C
C 0011 10 CONTINUE
C
C 0012 WRITE (WT, 100)
C 0013 100 FORMAT(///' Input NUMBER of Program to be displayed.',
C /' To EXIT, press RETURN, > ', $)
C 0014 READ (RT, 110) NUMBER
C 0015 110 FORMAT(I2)
C
C 0016 IF (NUMBER.EQ.0) RETURN
C 0018 IF (NUMBER.LT.1 .OR. NUMBER.GT.9) GOTO 400
C
C 0020 WRITE (WT, 112)
C 0021 112 FORMAT(///' Do you want output on printer? (Y or N) > ', $)
C 0022 READ (RT, 113) INPUT
C 0023 113 FORMAT(A1)
C 0024 IF (INPUT.EQ.'Y') OUTDV=6
C
C 0026 120 CONTINUE
C
C NOW TRANSMIT A 'REQUEST TO SAVE' TO ROBOT
C
C 0027 CALL REQTS(NUMBER, IERROR)
C
C 0028 IF (IERROR.EQ.1) GOTO 2100
C 0030 IF (IERROR.EQ.2) GOTO 2200
C 0032 IF (IERROR.GE.3) GOTO 2300
C
C NOW WAIT FOR SAVE ACKNOWLEDGE - TYPE CODE 66
C
C 0034 200 CONTINUE
C
C 0035 CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)
C
C 0036 IF (IERROR.EQ.1) GOTO 3100
C 0038 IF (IERROR.EQ.2) GOTO 3200
C 0040 IF (IERROR.GE.3) GOTO 3300
C
C 0042 ITYPE=IDATA(3)
C 0043 IACK=IDATA(4)
IPNUM=IDATA(5)
ISIZE=IDATA(6)+256*IDATA(7)

WRONG MESSAGE TYPE RECEIVED

IF (ITYPE.NE.66) GOTO 1100
IF (IPNUM.NE.NUMBER) GOTO 1200
NOT READY TO RECEIVE CODE = 0

IF (IACK.EQ.0) GOTO 1300
IF (IACK.GE.2) GOTO 1000

WRITE (OUTDV,202) NUMBER
FORMAT(' Robot Program No. ',I4,/) 202

IBLKCT=0

RECEIVES BLOCK NUMBER AND 252 BYTES OF DATA IN EACH BLOCK

ITOTBK=ISIZE/252
IREM=ISIZE-ITOTBK*252
IF (IREM.NE.0) ITOTBK=ITOTBK+1

LOOP TO RECEIVE AND PUT IN ARRAY Follows:

DO 300 I=1,ITOTBK
SKIP REQUEST TO SEND FOR FIRST BLOCK OF DATA
IF (I.EQ.1) GOTO 240
CALL REQTS(NUMBER,IERROR)
IF (IERROR.EQ.1) GOTO 2100
IF (IERROR.EQ.2) GOTO 2200
IF (IERROR.GE.3) GOTO 2300
CONTINUE

STARTS RECEIVING PROGRAM HERE

CALL RECEIVE(IERROR,ILEN,ICSUM,IDATA)
IF (IERROR.EQ.1) GOTO 3100
IF (IERROR.EQ.2) GOTO 3200
IF (IERROR.GE.3) GOTO 3300

ITYPE=IDATA(3)
IF (ITYPE.NE.67) GOTO 1100
IBLOCK=IDATA(4)
IF (IBLOCK.NE.IBLKCT) GOTO 4100

DISPLAY ARRAY ON CRT OR PRINTER

DO 250 K=5,LEN+2

IF(IDATA(K).EQ.'012) WRITE (OUTDV,205)

IF(IDATA(K).EQ.'012) WRITE (OUTDV,212)

IF(IBATA(K).EQ.0) GOTO 260

IF(IDATA(K).NE.'012) WRITE (OUTDV,220) IDATA(K)

FORMAT('A1,$)

CONTINUE

IBLKCT=IBLKCT+1

END OF FILE TRANSFER LOOP

CONTINUE

WRITE (OUTDV,310)

IF (OUTDV.EQ.6) CLOSE (UNIT=6)

GOTO 499
C------------------------------------------------------------------------
C          ALL OF THE VARIOUS ERROR MESSAGES FOLLOW:
C------------------------------------------------------------------------

0106  400 WRITE (WT,402)
0107  402 FORMAT(///,' ONLY PROGRAM NUMBERS 1-9 ACCEPTABLE.')
0108   GOTO 10

C

0109  499 CONTINUE  *
0110  500 WRITE (WT,500)
0111  500 FORMAT(///,' PRESS "RETURN" TO GOTO MAIN MENU.')
0112   READ (RT,904) INPUT
0113   RETURN

C

0114  900 CONTINUE
0115   RETURN

C

0116  904 READ (RT,904) INPUT
0117   FORMAT(A1)
0118   RETURN

C

0119 1000 WRITE (WT,1002)
0120 1002 FORMAT(///,' ILLEGAL ACKNOWLEDGE CODE RECEIVED.')
0121   GOTO 900

C

0122 1100 WRITE (WT,1102)
0123 1102 FORMAT(///,' WRONG MESSAGE TYPE RECEIVED!')
0124   GOTO 900

C

0125 1200 WRITE (WT,1202)
0126 1202 FORMAT(///,' WRONG PROGRAM NUMBER RETURNED BY ROBOT!')
0127   GOTO 900

C

0128 1300 WRITE (WT,1302)
0129 1302 FORMAT(///,' ROBOT NOT READY TO RECEIVE,')
           1     ' TYPE "RETURN" TO TRY AGAIN.')
0130   READ (RT,904) INPUT
0131   GOTO 120

C

0132 2100 WRITE (WT,2102)
0133 2102 FORMAT(///,' TIME OUT ERROR WHILE WAITING FOR',"
           1     " ACKNOWLEDGE FROM ROBOT.')
0134   GOTO 900

C

0135 2200 WRITE (WT,2202)
0136 2202 FORMAT(///,' LRC ERROR MESSAGE RETURNED AFTER TRANSMISSION.')
0137   GOTO 900

C

0138 2300 WRITE (WT,2302) IERROR
0139 2302 FORMAT(///,' ILLEGAL ACKNOWLEDGE CODE',I4,' RECEIVED',"
           1     ' AFTER TRANSMISSION TO ROBOT.')
0140 3100 WRITE (WT,3102)
0141 3102 FORMAT(///,' FIVE SECOND TIMEOUT OCCURRED DURING RECEIVE.')
FORTRAN IV  V02.5-2  Thu 08-May-86 13:24:38  PAGE 006

0142  GOTO 900  
C
0143  3200  WRITE (WT,3202)  
0144  3202  FORMAT(//,' CHECKSUM ERROR CALCULATION PRODUCED ERROR',  
1       ' DURING RECEIVE,' )  
0145  GOTO 900  
C
0146  3300  WRITE (WT,3302) IERROR  
0147  3302  FORMAT(//,' ILLEGAL ERROR CODE',I4,' RETURNED FROM',  
1       ' RECEIVE SUBROUTINE,' )  
0148  GOTO 900  
C
0149  4100  WRITE (WT,4102) IRLOCK,IBLKCT  
0150  4102  FORMAT(//,' BLOCK COUNT TRANSMITTED (',I4,')',  
1       ' DOES NOT MATCH EXPECTED BLOCK (',I4,').')  
0151  GOTO 900  
C
0152  END
FORTRAN IV  
Storage Map for Program Unit DISPLA

Local Variables, .PSECT $DATA, Size = 000054 ( 22, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILNO</td>
<td>I*2</td>
<td>000012</td>
<td>I</td>
<td>I*2</td>
<td>000044</td>
<td>IACK</td>
<td>I*2</td>
<td>000030</td>
</tr>
<tr>
<td>IBLKCT</td>
<td>I*2</td>
<td>000036</td>
<td>IBLOCK</td>
<td>I*2</td>
<td>000046</td>
<td>ICSUM</td>
<td>I*2</td>
<td>000024</td>
</tr>
<tr>
<td>IERROR</td>
<td>I*2</td>
<td>000020</td>
<td>ILEN</td>
<td>I*2</td>
<td>000022</td>
<td>INPUT</td>
<td>L*1</td>
<td>000016</td>
</tr>
<tr>
<td>IPNUM</td>
<td>I*2</td>
<td>000032</td>
<td>IREM</td>
<td>I*2</td>
<td>000042</td>
<td>isize</td>
<td>I*2</td>
<td>000034</td>
</tr>
<tr>
<td>ITOTBK</td>
<td>I*2</td>
<td>000040</td>
<td>ITYPE</td>
<td>I*2</td>
<td>000026</td>
<td>K</td>
<td>I*2</td>
<td>000050</td>
</tr>
<tr>
<td>LERROR</td>
<td>L*1</td>
<td>000017</td>
<td>NUMBER</td>
<td>I*2</td>
<td>000004</td>
<td>OUTDV</td>
<td>I*2</td>
<td>000014</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>000006</td>
<td>WT</td>
<td>I*2</td>
<td>000010</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block / /, Size = 003124 ( 810. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>000000</td>
<td>PROG</td>
<td>L*1</td>
<td>001004</td>
<td>BUFFER</td>
<td>L*1</td>
<td>003004</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section Offset</th>
<th>------</th>
<th>Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER</td>
<td>L*1</td>
<td>.$$$$</td>
<td>003004</td>
<td>000120</td>
<td>( 40.) (80)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>.$$$$</td>
<td>000000</td>
<td>001004</td>
<td>( 258.) (258)</td>
</tr>
<tr>
<td>PROG</td>
<td>L*1</td>
<td>.$$$$</td>
<td>001004</td>
<td>002000</td>
<td>( 512.) (1024)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td>REGTS</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE DDIR

ROUTINE TO DISPLAY THE DIRECTORY OF FILES LOCATED ON THE DK: DISKETTE.

NONE

READS AND DECODES DIRECTORY INFORMATION, DOES NOT AFFECT DISKETTE DIRECTORY.

DISPLAYS DK: DISKETTE DIRECTORY ON MINC CRT OR PRINTER.

CALLS TO: IGETC, LOOKUP, IREADW, ICLOSE, IFREEC, R50ASC, CLOSE

DISKETTE ON WHICH DDIR IS ATTEMPTED MUST HAVE HAD A SQUEEZE OPERATION AT LEAST ONCE SINCE ITS CREATION.

J. KEITH MCELVEEN /SOFTWARE DESIGN AND PROGRAMMED
JLM /NARRATION
JLM /REVISED NARRATION AND PROGRAM

(REVISE THIS NARRATIVE IF NECESSARY)
INTEGER*2 DBLK(4), RT, WT, IMTEMP, IDAY, IYEAR, IMNTH, CNT, B3

INTEGER*2 IDATA(2048), BUFFER, BUFF1(16), BUFF2(16)

INTEGER*2 INPUT

CREATE RAD50 FILE DESCRIPTOR FOR LOOKUP COMMAND

DATA DBLK /3RDK1, 3R, 3R, 3R /
DATA IBLK /6/

CREATE MONTH 3-LETTER DISPLAY TABLE TO BE INDEXED BY IMNTH,

INTEGER MNTH(13, 3)
DATA MNTH /'060', '112', '116', '115', '101', '115', '112', '112',
1 '101', '123', '117',
1 '116', '104', '060', '141', '145', '141', '160', '141', '165', '165',
1 '165', '145', '143',
1 '157', '145', '060', '156', '142', '162', '162', '171', '156', '154',
1 '147', '160', '164',
1 '166', '143'/

ASSIGN TERMINAL UNIT NUMBER

WT=7
RT=5
OUTDIV=7

K=1
LL=1
NS=0

WRITE(WT, 112)
112 FORMAT(/' DO YOU WANT OUTPUT ON PRINTER? (Y or N)> ')
113 READ(RT, 113) INPUT

IF (INPUT.EQ. 'Y') OUTDIV=6

OBTAIN CHANNEL NUMBER

ICHAN=IGETC()

IF (ICHAN.LT.0) GOTO 1060

LOOKUP DEVICE AT BLOCK 6

IF (LOOKUP(ICHAN, DBLK), LT, 0) GOTO 1050

READ (WITH WAIT OPTION) 2048 CHARACTERS STARTING AT BLOCK 6

**********MAKE LARGER THAN 2048 IF USING LARGE STORAGE
DEVICE. WILL NEED TO INCREASE IDATA DIMENSION TO NEW SIZE AND CHANGE ...

GOTO 100 IF END OF FILE ENCOUNTERED
GOTO 200 IF ERROR

ICODE=IREADW(2048,IDATA,IBLK,ICHAN)
IF (ICODE.EQ.-1) GOTO 100
IF (ICODE.LT.-1) GOTO 200

CLOSE AND FREE CHANNEL

CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)
SET NUMBER OF HIGHEST SEGMENT=NHS

NHS=IDATA(3)

SEARCH FOR STATUS WORDS:
'002000 PERMANENT FILE
'102000 PERMANENT PROTECTED FILE
'004000 END OF SEGMENT

IF(IDATA(K).EQ.'002000) GOTO 20
IF(IDATA(K).EQ.'102000) GOTO 20
IF(IDATA(K).EQ.'004000) NS=NS+1
IF(IDATA(K).EQ.'004000.AND.NHS.EQ.NS) GOTO 999

IF NOT A STATUS WORD INCREMENT K AND TRY AGAIN

K=K+1
GOTO 10

PUT NEXT THREE WORDS (1&2=FILENAME,3RD=EXTENSION IN RAD50 FORMAT) INTO TEMPORARY ARRAY AND CALL CONVERSION (SYSTEM) SUBROUTINE

CONTINUE
DO 30 L=1,3
BUFF1(L)=IDATA(K+L)
CONTINUE
BUFF1(4)=0
CALL R50ASC(10,BUFF1,BUFF2)

DECODE DATE WORD
K=K+6
IMNTH=MOD(IDATA(K),16384)
IMNTH=IMNTH/1024
IYEAR=MOD(IDATA(K),16)
IF(IYEAR.NE.0) IYEAR=IYEAR+72
IDAY=MOD(IDATA(K),1024)
IDAY=IDAY/32

OUTPUT (FILENAME, EXT LENGTH DATE) TO OUTPUT DEVICE
0058    IF (OUTDIV.EQ.6) WRITE(OUTDIV,300)(BUFF2(I),I=1,5),IDATA(K-2),
       1      IDAY,(MNTH(IMNTH+1,J),J=1,3),IYEAR
C
0060    300    FORMAT(+'$'5X,3A2,'.',2A2,2X,I4,2X,I2,'-',3A1,'-',I2,5X,$)
C
0061    IF (OUTDIV.EQ.7) WRITE(OUTDIV,301)(BUFF2(I),I=1,5),IDATA(K-2),
       1      IDAY,(MNTH(IMNTH+1,J),J=1,3),IYEAR
C
0063    301    FORMAT(+'$',5X,3A2,'.',2A2,2X,I4,2X,I2,'-',3A1,'-',I2,5X,$)
C
0064    IF(LL.EQ.2) WRITE(OUTDIV,310)
0066    310    FORMAT(+'$'
0067    LL=LL+1
0068    IF(LL.EQ.3) LL=1
0070    GOTO 10
C
0071    999    CONTINUE
C
0072    IF (OUTDIV.EQ.6) CLOSE(UNIT=6)
C
0074    GOTO 1000
C
0075    100    WRITE(WT,110)
0076    110    FORMAT(+'$' END OF FILE ON READ')
0077    GOTO 1000
0078    200    WRITE(WT,210)
0079    210    FORMAT(+'$' ERROR ON READ')
C
0080    1000    WRITE (WT,1010)
0081    1010    FORMAT(+'$' TO CONTINUE PRESS RETURN> '$')
0082    READ (RT,1020) INPUT
0083    1020    FORMAT(A2)
0084    RETURN
C
0085    1050    CONTINUE
0086    WRITE (WT,1052)
0087    1052    FORMAT(+'$' BAD LOOKUP ERROR')
0088    GOTO 1000
C
0089    1060    CONTINUE
0090    WRITE (WT,1062)
0091    1062    FORMAT(+'$' CANNOT ALLOCATE CHANNEL ERROR')
0092    GOTO 1000
C
0093    END
FORTRAN IV Storage Map for Program Unit DDIR

Local Variables: .PSECT $DATA, Size = 010332 (2157, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER</td>
<td>I*2</td>
<td>010256</td>
<td>B3</td>
<td>I*2</td>
<td>010254</td>
<td>CNT</td>
<td>I*2</td>
<td>010252</td>
</tr>
<tr>
<td>I</td>
<td>I*2</td>
<td>010304</td>
<td>IBLK</td>
<td>I*2</td>
<td>010226</td>
<td>ICHAN</td>
<td>I*2</td>
<td>010274</td>
</tr>
<tr>
<td>ICODE</td>
<td>I*2</td>
<td>010276</td>
<td>IDAY</td>
<td>I*2</td>
<td>010244</td>
<td>IMNTH</td>
<td>I*2</td>
<td>010250</td>
</tr>
<tr>
<td>INPUT</td>
<td>I*2</td>
<td>010260</td>
<td>ITEMP</td>
<td>I*2</td>
<td>010242</td>
<td>IYEAR</td>
<td>I*2</td>
<td>010246</td>
</tr>
<tr>
<td>J</td>
<td>I*2</td>
<td>010306</td>
<td>K</td>
<td>I*2</td>
<td>010266</td>
<td>L</td>
<td>I*2</td>
<td>010302</td>
</tr>
<tr>
<td>LL</td>
<td>I*2</td>
<td>010270</td>
<td>NHS</td>
<td>I*2</td>
<td>010300</td>
<td>NS</td>
<td>I*2</td>
<td>010272</td>
</tr>
<tr>
<td>OUTDIV</td>
<td>R*4</td>
<td>010262</td>
<td>RT</td>
<td>I*2</td>
<td>010236</td>
<td>WT</td>
<td>I*2</td>
<td>010240</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFF1</td>
<td>I*2</td>
<td>$DATA</td>
<td>010010</td>
<td>000040</td>
<td>(16) (16)</td>
</tr>
<tr>
<td>BUFF2</td>
<td>I*2</td>
<td>$DATA</td>
<td>010050</td>
<td>000040</td>
<td>(16) (16)</td>
</tr>
<tr>
<td>DBLK</td>
<td>I*2</td>
<td>$DATA</td>
<td>000000</td>
<td>000010</td>
<td>(4) (4)</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000010</td>
<td>010000</td>
<td>(2048) (2048)</td>
</tr>
<tr>
<td>MNTH</td>
<td>I*2</td>
<td>$DATA</td>
<td>010110</td>
<td>000116</td>
<td>(39) (13,3)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
<td>IREADW</td>
<td>I*2</td>
</tr>
<tr>
<td>MOD</td>
<td>I*2</td>
<td>R50ASC</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

144
MODULE NAME: SUBROUTINE LIST

PURPOSE: ROUTINE TO LIST ANY SELECTED PROGRAM STORED ON THE DK: DISKETTE

INPUT: ACCEPTS PROGRAM NAME FROM MINC KEYBOARD AND THEN LOADS SELECTED PROGRAM INTO ARRAY IN MEMORY.

PROCESSING: REQUIRES FORMATTING AND PLACING '/' AT END OF EACH LINE TO GIVE SAME APPEARANCE AS ROBOT CONSOLE DISPLAY.

OUTPUT: DISPLAYS PROGRAM ON MINC CRT OR PRINTER.

CALLED BY: CYR02

CALLS TO: DISKRD, SCOPY, INSERT, IRAD50, IGETC, IFETCH, LOOKUP, ICLOSE, IFREEC, IREAD, CLOSE

SPECIAL INTERFACE REQUIREMENTS: NONE

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER CONTACT/ACTION/REMARKS
6-29-85 FRED R. SIAS, JR. (803)-656-3375/DESIGN & PROGRAM
7-22-85 J. KEITH MCELVEEN (803)-656-3375/REVISED PROGRAM/NARR
INTEGER*2 OUTDV
BYTE INPUT, FILNAM(15), NAMFIL(7), ARRAY(5000), IBUFF(512)
LOGICAL*1 LERROR
REAL*8 FILE

COMMON /INOUT/ ARRAY, BPDATA

CALL SCOPY('DY1, CYR,FILNAM)
DATA FILNAM(13) /00/
DATA NAMFIL(7) /00/

WT=7
RT=5

CONTINUE
OUTDV=7

WRITE (WT,100)
FORMAT ('//,' Input FILE NAME of program to list.'')

READ (RT,105) (NAMFIL(I), I=1,6)
FORMAT (6A1)

IF (NAMFIL(1).EQ.' ') RETURN
WRITE (WT,107)
FORMAT ('//,' Do you want output on printer? (Y or N) ')
READ (RT,108) INPUT

IF (INPUT.EQ.'Y') OUTDV=6
CONTINUE

CALL INSERT(NAMFIL, FILNAM, 4, 6)

CALL IRAD50(12,FILNAM, FILE)

OPEN FILE AND READ

ICHAN=IGETC()
IF (ICHAN.LT.0) STOP 'NO CHANNEL AVAILABLE'
IERROR=IFETCH(FILE)
IF (IERROR.NE.0) STOP 'BAD FETCH'
IERROR=LOOKUP(ICHAN, FILE)
IF (IERROR.EQ.-2) GOTO 450
IF (IERROR.LT.0, ANB, IERROR, NE.-2) STOP 'BAD LOOKUP'

WRITE (OUTDV,170) (FILNAM(I), I=1,12)
FORMAT (' Diskette program name: ')
IBLOCK=0
INDEX=1
CONTINUE

C------------------------READ DATA
C
IERROR=IREADW(256, IBUFF, IBLOCK, ICHAN)
IF (IERROR.LT.-1) STOP 'BAD READ'
IF (IERROR.EQ.-1) GOTO 140
DO 135 I=1,512
ARRAY(INDEX)=IBUFF(I)
INDEX=INDEX+1
CONTINUE
IBLOCK=IBLOCK+1
GOTO 130
CONTINUE

CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)

GET PROGRAM-LENGTH
DO 150 I=1,5000
IF (ARRAY(I).EQ.0) GOTO 160
CONTINUE
ISIZE=I-1
DO 200 I=1,ISIZE
IF (ARRAY(I).EQ.015) WRITE (OUTDV,182)
FORMAT('+','/',$)
WRITE (OUTDV,189) ARRAY(I)
FORMAT('+','A1',$)
CONTINUE
IF (OUTDV.EQ.6) CLOSE (UNIT=6)
WRITE (WT,500)
FORMAT(//' PRESS 'RETURN' TO CONTINUE> ',$)
READ (RT,510) INPUT
FORMAT(A1)
RETURN
CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)
IF (OUTDIV.EQ.6) CLOSE(UNIT=6)
WRITE (WT,455)
FORMAT(//' FILE NOT FOUND. PRESS "RETURN" TO TRY AGAIN> ',$)
READ(RT,510) INPUT
GOTO 10
END
FORTRAN IV  Storage Map for Program Unit LIST

Local Variables: .PSECT $DATA, Size = 001144 (306, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>R*8</td>
<td>001044</td>
<td>I</td>
<td>I*2</td>
<td>001064</td>
</tr>
<tr>
<td>ICHAN</td>
<td>I*2</td>
<td>001066</td>
<td>IERROR</td>
<td>I*2</td>
<td>001070</td>
</tr>
<tr>
<td>INPUT</td>
<td>L*1</td>
<td>001042</td>
<td>ISIZE</td>
<td>I*2</td>
<td>001076</td>
</tr>
<tr>
<td>OUTDIV</td>
<td>R*4</td>
<td>001100</td>
<td>OUTDV</td>
<td>I*2</td>
<td>001040</td>
</tr>
<tr>
<td>WT</td>
<td>R*4</td>
<td>001054</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /INOUT/, Size = 011610 (2502, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>000000</td>
</tr>
<tr>
<td>BPDATA</td>
<td>R*4</td>
<td>011610</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>INOUT</td>
<td>000000</td>
<td>011610</td>
<td>(2500) (5000)</td>
</tr>
<tr>
<td>FILNAM</td>
<td>L*1</td>
<td>$DATA</td>
<td>000000</td>
<td>000017</td>
<td>(8) (15)</td>
</tr>
<tr>
<td>IBUFF</td>
<td>L*1</td>
<td>$DATA</td>
<td>000026</td>
<td>001000</td>
<td>(256) (512)</td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>$DATA</td>
<td>000017</td>
<td>000007</td>
<td>(4) (7)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>I*2</td>
<td>IFETCH</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
</tr>
<tr>
<td>IRAD50</td>
<td>I*2</td>
<td>IREADW</td>
<td>I*2</td>
<td>LOOKUP</td>
<td>I*2</td>
<td>SCOPY</td>
<td>R*4</td>
</tr>
</tbody>
</table>
SUBROUTINE RESEQ

THIS SUBROUTINE ACCEPTS THE NAME OF A ROBOT PROGRAM STORED ON DISKETTE AND THEN RENUMBERS THE
COMMAND LINES SO THAT THE PROGRAM STARTS WITH TWO COMMENT LINES FOLLOWED BY PROGRAM LINE NUMBERS
IN STEPS OF TEN.

INPUT: 
ACCEPTS PROGRAM FILE NAME FROM THE MINC KEYBOARD AND THEN READS THAT PROGRAM FROM THE DISKETTE.
PROGRAM ASSUMES *.CYR FILENAME EXTENSION.

PROCESSING:

OUTPUT: 
REWRITES THE RESEQUENCED PROGRAM TO DISKETTE.

CALLED BY: 
CYRO2

CALLS TO: 
INSRT, IRAD50, SCOPY, INSRT, IGETC, IFETCH, LOOKUP, IREADW, ICLOSE, IWRITW

SPECIAL INTERFACE REQUIREMENTS: 
NONE

REVISION HISTORY 
(REVISE THIS NARRATIVE IF NECESSARY)

DATE 
PROGRAMMER 
==== CONTACT/ACTION/REMARKS ====
7-20-85 
J. KEITH MCELVEEN (803)656-3375/SYSTEM DESIGN/PROGRAM
7-22-85 
JKM (803)656-3375/REVISED NARRATION
5-07-86 
FRS INCREASED ARRAY SIZE
BYTE INPUT,FILNAM(15),NAMFIL(7),ARRAY(8000),
1 IBUFF(512),ARRAY1(8000)
LOGICAL*1 LERROR
REAL*8 FILE
COMMON/INSRT/I,J,K,L,ARRAY1,ARRAY,LNUM
CALL SCOPY('DY1 CYR',FILNAM)
DATA FILNAM(13) /00/
DATA NAMFIL(7) /00/
WT=7
RT=5
CONTINUE
WRITE (WT,100)
FORMAT ('//,' 'Input FILE NAME of program to RESEQUENCE,',
1 '/,' To EXIT, press RETURN> ')$
READ (RT,105) (NAMFIL(I),I=1,6)
FORMAT(6A1)
IF (NAMFIL(1).EQ,' ') RETURN
CONTINUE
CALL INSERT(NAMFIL,FILNAM,4,6)
CALL IRAD50(12,FILNAM,FILE)
WRITE (WT,170) (FILNAM(I),I=1,12)
FORMAT(' Diskette program name: ',12A1,//)
OPEN FILE AND READ
ICHAN=IGETC()
IF (ICHAN.LT.0) STOP 'NO CHANNEL AVAILABLE'
IERROR=IFETCH(FILE)
IF (IERROR.NE.0) STOP 'BAD FETCH'
IERROR=LOOKUP(ICHAN,FILE)
IF (IERROR.EQ.-2) GOTO 450
IF (IERROR.LT.0.AND.IERROR.NE.-2) STOP 'BAD LOOKUP'
IBLOCK=0
INDEX=1
CONTINUE
OPEN FILE AND READ
IERROR=IREADW(256,IBUFF,IBLOCK,ICHAN)
IF (IERROR.LT.-1) STOP 'BAD READ'
IF (IERROR.EQ.-1) GOTO 140
DO 135 I=1,512
C 135
C 140
C
C 0043  ARRAY(INDEX)=IBUFF(I)
C 0044   INDEX=INDEX+1
C
C 0045   IF (INDEX.GE.8000) GOTO 2000
C
C 0047   CONTINUE
C 0048   IBLOCK=IBLOCK+1
C 0049   GOTO 130
C 0050   CONTINUE
C-------------------------------------------------------
C
C 0051   CALL ICLOSE(ICHAN)
C 0052   CALL IFREEC(ICHAN)
C
C 0053   I=1
C 0054   J=1
C 0055   LNUM=0
C
C 0056   L=0
C 0057   IF(ARRAY(1).NE."116) GOTO 900
C 0059   CALL INSRT
C 0060   J=J+5
C 0061   I=I+K-1
C
C 0062   SET FIRST TIME INDICATOR(L) TO 1
C
C 0063  700   J=J+1
C 0064   I=I+1
C 0065   IF(ARRAY(I).EQ.0) GOTO 800
C
C 0066   IF(ARRAY(I).EQ."012.AND.ARRAY(I+1).EQ."116) GOTO 710
C 0069   ARRAY1(J)=ARRAY(I)
C 0070   GOTO 700
C 0071  710   CALL INSRT
C 0072   J=J+5
C 0073   I=I+K-1
C 0074   GOTO 700
C 0075  800   CONTINUE
C*******************************************7MAY86
C
C 0076   PUT ZERO AT END OF ARRAY
C 0077   ARRAY(J)=0
C 0077   ARRAY(J+1)=0
C
C 0078   CALCULATED NUMBER OF WORDS TO OUTPUT
C
C 0079  900   J=J-1
C 0079   JJ=J/2+1
C*******************************************7MAY86
C
C 0080   OUTPUT TO DISK

151
ICHAN = IGETC()
IF (ICHAN .LT. 0) STOP 'NO CHANNEL AVAILABLE'
IERROR = IFETCH(FILE)
IF (IERROR .NE. 0) STOP ' BAD FETCH'
IERROR = LOOKUP(ICHAN, FILE)
IF (IERROR .LT. 0) STOP ' BAD LOOKUP'
IERROR = IWRITE(JJ, ARRAY1, 0, ICHAN)

CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)
GOTO 920
WRITE(WT, 910)
FORMAT(//, 'FIRST CHARACTER IN FILE TO BE RESEQUENCED
1 IS NOT AN 'N', ',', 'CHECK FILE LISTING,')
CONTINUE
WRITE(WT, 930)
FORMAT(//, 'Press RETURN to continue> ', $)
READ (RT, 904) INPUT
RETURN
CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)
WRITE(WT, 455)
FORMAT(//, 'FILE NOT FOUND. PRESS 'RETURN' TO CONTINUE> ', $)
READ (RT, 460) INPUT
GOTO 10
CONTINUE
CALL ICLOSE(ICHAN)
CALL IFREEC(ICHAN)
WRITE(WT, 2010)
FORMAT(//, 'PROGRAM 'F!4Al', TOO LARGE TO RESEQUENCE',//,
'PRESS 'RETURN' TO CONTINUE> ', $)
READ (RT, 460) INPUT
END
FORTRAN IV
Storage Map for Program Unit RESEQ

Local Variables, .PSECT $DATA, Size = 001141 (305, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>R*8</td>
<td>001036</td>
<td>IERROR</td>
<td>I*2</td>
<td>001062</td>
<td>ICHAN</td>
<td>I*2</td>
<td>001056</td>
</tr>
<tr>
<td>JJ</td>
<td>I*2</td>
<td>001060</td>
<td>INDEX</td>
<td>I*2</td>
<td>001064</td>
<td>INPUT</td>
<td>L*1</td>
<td>001034</td>
</tr>
<tr>
<td>WT</td>
<td>R*4</td>
<td></td>
<td>LERROR</td>
<td>L*1</td>
<td>001035</td>
<td>RT</td>
<td>R*4</td>
<td>001052</td>
</tr>
</tbody>
</table>

COMMON Block /INSRT /, Size = 037212 (8005, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>000000</td>
<td>J</td>
<td>I*2</td>
<td>000002</td>
<td>K</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>L</td>
<td>I*2</td>
<td>000006</td>
<td>ARRAY1</td>
<td>L*1</td>
<td>000010</td>
<td>ARRAY</td>
<td>L*1</td>
<td>017510</td>
</tr>
<tr>
<td>LNUM</td>
<td>I*2</td>
<td>037210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>INSRT</td>
<td>017510</td>
<td>017500</td>
<td>(4000) (8000)</td>
</tr>
<tr>
<td>ARRAY1</td>
<td>L*1</td>
<td>INSRT</td>
<td>000010</td>
<td>017500</td>
<td>(4000) (8000)</td>
</tr>
<tr>
<td>FILNAM</td>
<td>L*1</td>
<td>$DATA</td>
<td>000000</td>
<td>000017</td>
<td>(8) (15)</td>
</tr>
<tr>
<td>IBUFF</td>
<td>L*1</td>
<td>$DATA</td>
<td>000026</td>
<td>001000</td>
<td>(256) (512)</td>
</tr>
<tr>
<td>NAMFIL</td>
<td>L*1</td>
<td>$DATA</td>
<td>000017</td>
<td>000007</td>
<td>(4) (7)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>I*2</td>
<td>IFETCH</td>
<td>I*2</td>
<td>IFREEC</td>
<td>I*2</td>
<td>IGETC</td>
<td>I*2</td>
<td>INSERT</td>
<td>I*2</td>
</tr>
<tr>
<td>INSRT</td>
<td>I*2</td>
<td>IRAD50</td>
<td>I*2</td>
<td>IREADW</td>
<td>I*2</td>
<td>IWRITE</td>
<td>I*2</td>
<td>LOOKUP</td>
<td>I*2</td>
</tr>
<tr>
<td>SCOPY</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE INSRT

TO INSERT NEW RESEQUENCED LINE NUMBERS

CALLED BY: RESEQ

CALLS TO: NONE

DATE
7-22-85

J. KEITH MCELVEEN
ROUTINE WRITTEN

COMON/INSRT/I, J, K, L, ARRAY1, ARRAY, LNUM
BYTE ARRAY1(8000), ARRAY(8000), IASC(10)

LNUM=LNUM+1

ARRAY1(J)=ARRAY(I)
IF(L.EQ.0) J=0
IF(L.EQ.0) 1=0
ARRAY1(J+1)=ARRAY(I+1)

K=1

IF(ARRAY(I+K),GE,'060, AND, ARRAY(I+K),LE,'071) GOTO 800
I1=INT(LNUM/100.)
I2=-1*I1*10+INT(LNUM/10.)
I3=-1*(I1*100+I2*10)+LNUM

ARRAY1(J+2)=IASC(I1+1)
ARRAY1(J+3)=IASC(I2+1)
ARRAY1(J+4)=IASC(I3+1)
ARRAY1(J+5)=*060

RETURN
END
FORTRAN IV Storage Map for Program Unit INSRT

Local Variables, .PSECT $DATA, Size = 000031 ( 13. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I*2</td>
<td>000012</td>
<td>I2</td>
<td>I*2</td>
<td>000014</td>
<td>I3</td>
<td>I*2</td>
<td>000016</td>
</tr>
</tbody>
</table>

COMMON Block /INSRT /, Size = 037212 ( 8005. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>000000</td>
<td>J</td>
<td>I*2</td>
<td>000002</td>
<td>K</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>L</td>
<td>I*2</td>
<td>000006</td>
<td>ARRAY1</td>
<td>L*1</td>
<td>000010</td>
<td>ARRAY</td>
<td>L*1</td>
<td>017510</td>
</tr>
<tr>
<td>LNUM</td>
<td>I*2</td>
<td>037210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>L*1</td>
<td>INSRT 017510</td>
<td>017500</td>
<td>(4000,) (8000)</td>
</tr>
<tr>
<td>ARRAY1</td>
<td>L*1</td>
<td>INSRT 000010</td>
<td>017500</td>
<td>(4000,) (8000)</td>
</tr>
<tr>
<td>IASC</td>
<td>L*1</td>
<td>$DATA 000000</td>
<td>000012</td>
<td>(5,) (10)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>I*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE MESSAGE

PURPOSE:
THIS ROUTINE ACCEPTS A MESSAGE FROM COMPUTER KEYBOARD
AND TRANSMITS TO ROBOT FOR DISPLAY ON CONSOLE

INPUT:
ACCEPTS MESSAGE FROM MINC KEYBOARD.

PROCESSING:
SETS UP MESSAGE IN ARRAY AND HANDLES
HANDSHAKING AND TRANSMISSION OF MESSAGE.

OUTPUT:
TRANSMITS MESSAGE TO ROBOT WHERE IT IS
DISPLAYED ON ROBOT CONSOLE.

CALLED BY:
CYRO2

CALLS TO:
GETLIN, SEND

SPECIAL INTERFACE REQUIREMENTS:
USES MINC DIGITAL I/O MODULES.

REVISION HISTORY

<table>
<thead>
<tr>
<th>DATE</th>
<th>PROGRAMMER</th>
<th>CONTACT/ACTION/REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-24-84</td>
<td>FRED R. SIAS, JR.</td>
<td>(803)-656-3375/SYSTEM DESIGN/PROGRAM</td>
</tr>
<tr>
<td>6-04-85</td>
<td>FRED R. SIAS, JR.</td>
<td>/REVISED NARRATION</td>
</tr>
</tbody>
</table>
BYTE STRING(254)
INTEGER*2 IDATA(257), RT, WT, INPUT
LOGICAL*1 PROMT(16)

DATA PROMT /'I','N','P','U','T','S','E','00/
RT=5
WT=7
IDATA(2)=1
IDATA(3)=133

CALL GTLIN(STRING,PROMT)
LENGTH=LEN(STRING)
ILEN=LENGTH+3
DO 100 I=4, ILEN
IDATA(I)=STRING(I-3)
CONTINUE
IDATA(1)=LENGTH-H
CALL SEND(IERROR, ILEN, IDATA)

IF (IERROR.EQ.2) WRITE (WT,110)
FORMAT(//'CHECKSUM ERROR, DATA MAY BE WRONG.')
IF (IERROR.EQ.1) WRITE (WT,120)
FORMAT(//'TIMEOUT ERROR. ROBOT DID NOT RESPOND.')
WRITE (WT,130)
FORMAT(//'TYPE 'RETURN' TO CONTINUE.')
READ (RT,140) INPUT
FORMAT(14)

RETURN
END
Local Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>001436</td>
</tr>
<tr>
<td>INPUT</td>
<td>I*2</td>
<td>001430</td>
</tr>
<tr>
<td>WT</td>
<td>I*2</td>
<td>001426</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000376</td>
<td>257</td>
<td>(257)</td>
</tr>
<tr>
<td>PROMT</td>
<td>L*1</td>
<td>$DATA</td>
<td>001400</td>
<td>8</td>
<td>(16)</td>
</tr>
<tr>
<td>STRING</td>
<td>L*1</td>
<td>$DATA</td>
<td>000000</td>
<td>127</td>
<td>(254)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTLIN</td>
<td>R*4</td>
<td>LEN</td>
<td>I*2</td>
<td>SEND</td>
<td>R*4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE POSIT

PURPOSE:

THIS ROUTINE REQUESTS THE CURRENT ROBOT POSITION AND
AND DISPLAYS ON MINC CRT.

INPUT:    NONE

PROCESSING:

THE ROUTINE TRANSMITS A REQUEST FOR ROBOT POSITIONS
(TYPE CODE 132) WITH THE SINGLE BYTE MESSAGE ZERO (0) THAT
INDICATES THAT ONLY ONE RESPONSE PER REQUEST IS DESIRED.
THEN THE ROUTINE WAITS FOR A RESPONSE FROM THE ROBOT WHICH
SHOULD INCLUDE THE TYPE CODE 4 FOLLOWED BY EIGHTEEN (18)
BYTES OF BINARY DATA THAT CONVEY NINE CURRENT ROBOT POSITIONS.
EACH POSITION CONSISTS OF TWO SEQUENTIAL BYTES, LOW-ORDER
BYTE FIRST FOLLOWED BY THE HIGH-ORDER BYTE.

OUTPUT:    DISPLAYS CURRENT ROBOT POSITION

CALLED BY: CYR02

CALLS TO:

SEND (IERROR,ILN,IDATA)

RECEIVE (IERROR,ILN,ICSUM,IDATA)

SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULES.

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE          PROGRAMMER        CONTACT/ACTION/REMARKS
6-26-84        FRED R. SIAS, JR.     (803)-656-3375/SYSTEM DESIGN
10-29-84       FRED R. SIAS, JR.    /REV. POSITION INTEGER TO REAL
6-4-85         FRED R. SIAS, JR.   /REVISED NARRATION

159
BYTE STRING(254), INPUT
REAL*4 IXDATA, IYDATA, IZDATA, ICXPOS, ICYPOS, IDXPOS, IDYPOS, DATA
INTEGER*2 IDATA(257)
LOGICAL*1 PROMT(16)

RT=5
WT=7

LENGTH OF MESSAGE
IDATA(1)=2

SETS SEQUENCE NO. TO ZERO
IDATA(2)=0

SETS TYPE CODE
IDATA(3)=132

SETS RATE TO ONCE PER REQUEST
IDATA(4)=0

NO OF BYTES TO TRANSMIT IN SEND
ILEN=4

CALL SEND(IERROR, ILEN, IDATA)

IF (IERROR, EQ, 0) GOTO 5

WRITE (WT, 2) IERROR
FORMAT( 'ERROR CODE', I4, ' RECEIVED FROM SUBR. SEND.' )

CONTINUE

CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)

IF (IERROR, EQ, 2) WRITE (WT, 10)
FORMAT( 'CHECKSUM ERROR, DATA MAY BE WRONG!' )

WRITE (WT, 12)
FORMAT( 'PRESS RETURN TO CONTINUE' )
READ (RT, 210) INPUT

SHOULD BE MESSAGE TYPE CODE 4

IF (IDATA(3), NE, 4) WRITE (WT, 15) IDATA(3)
FORMAT( 'ILLEGAL TYPE CODE', I2, ' RECEIVED' )

IXDATA=IDATA(5)
DATA=IDATA(4)
IXDATA=(IXDATA*256+,DATA)/64.

WRITE (WT, 100) IXDATA
FORMAT( 'X-axis position is ', F6.2, ' inches.' )
FORTRAN IV  V02.5-2  Thu 08-May-86 13:31:07  PAGE 003

0031  IYDATA = IDATA(7)
0032  IYDATA = (IYDATA*256 + IDATA(6))/64.
0033  WRITE (WT,110) IYDATA
0034  110  FORMAT(' Y-axis position is ','F6.2',' inches.')
0035  IZDATA = IDATA(9)
0036  IZDATA = (IZDATA*256 + IDATA(8))/64.
0037  WRITE (WT,120) IZDATA
0038  120  FORMAT(' Z-axis position is ','F6.2',' inches.')
0039  ICXPOS = IDATA(11)
0040  ICXPOS = (ICXPOS*256 + IDATA(10))/10.
0041  WRITE (WT,130) ICXPOS
0042  130  FORMAT(' X-axis is ','F6.2',' degrees.')
0043  ICYPOS = IDATA(13)
0044  ICYPOS = (ICYPOS*256 + IDATA(12))/10.
0045  WRITE (WT,140) ICYPOS
0046  140  FORMAT(' Y-axis is ','F6.2',' degrees.')
0047  IDXPOS = IDATA(15)
0048  IDXPOS = (IDXPOS*256 + IDATA(14))/10.
0049  WRITE (WT,150) IDXPOS
0050  150  FORMAT(' C positioner, X-axis is ','F6.2',' degrees.')
0051  IDYPOS = IDATA(17)
0052  IDYPOS = (IDYPOS*256 + IDATA(16))/10.
0053  WRITE (WT,160) IDYPOS
0054  160  FORMAT(' C positioner, Y-axis is ','F6.2',' degrees.')
0055  IDXPOS = IDATA(19)
0056  IDXPOS = (IDXPOS*256 + IDATA(18))/10.
0057  WRITE (WT,170) IDXPOS
0058  170  FORMAT(' D positioner, X-axis is ','F6.2',' degrees.')
0059  IDYPOS = IDATA(21)
0060  IDYPOS = (IDYPOS*256 + IDATA(20))/10.
0061  WRITE (WT,180) IDYPOS
0062  180  FORMAT(' D positioner, Y-axis is ','F6.2',' degrees.')
0063  WRITE (WT,200)
0064  200  FORMAT('/// CARRIAGE RETURN TO CONTINUE ',$)
0065  READ (RT,210) INPUT
0066  210  FORMAT(I4)
0067  RETURN
0068  END

161
Local Variables, .PSECT $DATA; Size = 001534 ( 430, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>R*4</td>
<td>001456</td>
<td>ICYPOS</td>
<td>R*4</td>
<td>001442</td>
<td>ICXPOS</td>
<td>R*4</td>
<td>001436</td>
</tr>
<tr>
<td>ICYPOS</td>
<td>R*4</td>
<td>001442</td>
<td>IDXPOS</td>
<td>R*4</td>
<td>001446</td>
<td>IDYPOS</td>
<td>R*4</td>
<td>001452</td>
</tr>
<tr>
<td>IERROR</td>
<td>I*2</td>
<td>001474</td>
<td>ILEN</td>
<td>I*2</td>
<td>001472</td>
<td>INPUT</td>
<td>L*1</td>
<td>001420</td>
</tr>
<tr>
<td>IYDATA</td>
<td>R*4</td>
<td>001422</td>
<td>IYDATA</td>
<td>R*4</td>
<td>001426</td>
<td>I2DATA</td>
<td>R*4</td>
<td>001432</td>
</tr>
<tr>
<td>RT</td>
<td>R*4</td>
<td>001462</td>
<td>WT</td>
<td>R*4</td>
<td>001466</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA</td>
<td>000376</td>
<td>001002</td>
<td>(257) (257)</td>
</tr>
<tr>
<td>PROMT</td>
<td>L*1</td>
<td>$DATA</td>
<td>001400</td>
<td>000020</td>
<td>(8) (16)</td>
</tr>
<tr>
<td>STRING</td>
<td>L*1</td>
<td>$DATA</td>
<td>000000</td>
<td>000376</td>
<td>(127) (254)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td>SEND</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE PARS

 PURPOSE:

 THIS ROUTINE REQUESTS THE CURRENT ROBOT PARAMETERS AND
 AND DISPLAYS THEM ON MINC CRT.

 INPUT:   NONE

 PROCESSING: SETS UP REQUEST MESSAGE AND TRANSMITS.
 SCALES DATA RETURNED AND FORMATS.

 OUTPUT:   DISPLAYS PARAMETERS ON THE MINC CRT.

 CALLED BY: CYRO1

 CALLS TO: SEND, RECEIVE

 SPECIAL INTERFACE REQUIREMENTS: USES MINC DIGITAL I/O MODULES.

 REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

 DATE    PROGRAMMER   CONTACT/ACTION/REMARKS

 6-27-84  FRED R. SIAS, JR.  (803)-656-3375/DESIGN AND PROGRAM
 6-04-85  FRED R. SIAS, JR.  /REVISED NARRATION
BYTE STRING(254)
INTEGER*2 IDATA(257)
REAL*4 IXDATA, IYDATA, IZDATA, ICXPOS
LOGICAL*1 PROMT(16)

RT=5
WT=7

SETS SEQUENCE NO. TO ZERO
IDATA(2)=0

REQUEST CODE FOR SYSTEM PARAMETERS
IDATA(3)=138
IDATA(1)=2
IDATA(4)=0

ILEN=4
CALL SEND(IERROR, ILEN, IDATA)

NOW WAIT FOR MESSAGE TO COME BACK WITH INFO
CALL RECEIVE(IERROR, ILEN, ICSUM, IDATA)

IF (IERROR.EQ.2) WRITE (WT, 10)
FORMAT(‘CHECKSUM ERROR. DATA MAY BE WRONG!’)

IF (IDATA(3),NE,7) WRITE (WT, 15) IDATA(3)
FORMAT(‘ILLEGAL TYPE CODE ‘,I2,’ RECEIVED’)

IXDATA=IDATA(5)
IXDATA=(IXDATA*256+IDATA(4))/64.
WRITE (WT, 101)
FORMAT(4H0***,//)
WRITE (WT, 100) IXDATA

IYDATA=IDATA(7)
IYDATA=(IYDATA*256+IDATA(6))/64.
WRITE (WT, 110) IYDATA

IZDATA=IDATA(9)
IZDATA=IZDATA*256+IDATA(8)

ONE BIT EQUALS .1 PERCENT
IZDATA=IZDATA/10.
WRITE (WT, 120) IZDATA

ICXPOS=IDATA(11)
ICXPOS = ICXPOS * 256 + IDATA(10)

C ONE BIT EQUALS .1 PERCENT

0039 ICXPOS = ICXPOS / 10,

C WRITE (WT, 130) ICXPOS

0040 130 FORMAT (' AVC/ACC SetPoint Level ', F6.2, ' percent,...

C

0041 IF (IDATA(12) .EQ. 0) WRITE (WT, 140)

0043 IF (IDATA(12) .EQ. 1) WRITE (WT, 150)

0045 IF (IDATA(12) .EQ. 2) WRITE (WT, 160)

0047 IF (IDATA(12) .GT. 2) WRITE (WT, 170) IDATA(12)

C

0049 140 FORMAT (' No Oscillation has occurred. ')

0050 150 FORMAT (' Left Oscillation has occurred. ')

0051 160 FORMAT (' Right Oscillation has occurred. ')

0052 170 FORMAT (' ILLEGAL OSCILLATION CODE ', I8, ' RECEIVED. ')

C

0053 WRITE (WT, 200)

0054 200 FORMAT ('/,' CARRIAGE RETURN TO CONTINUE> ', $)

0055 READ (RT, 210) INPUT

0056 210 FORMAT (I4)

0057 RETURN

0058 END
Local Variables, .PSECT $DATA, Size = 001522 (425, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICSUM</td>
<td>I*2</td>
<td>001454</td>
<td>ICXPOS</td>
<td>R*4</td>
<td>001434</td>
<td>IERROR</td>
<td>I*2</td>
<td>001452</td>
</tr>
<tr>
<td>ILEN</td>
<td>I*2</td>
<td>001450</td>
<td>INPUT</td>
<td>I*2</td>
<td>001456</td>
<td>IXDATA</td>
<td>R*4</td>
<td>001420</td>
</tr>
<tr>
<td>IYDATA</td>
<td>R*4</td>
<td>001424</td>
<td>IZDATA</td>
<td>R*4</td>
<td>001430</td>
<td>RT</td>
<td>R*4</td>
<td>001440</td>
</tr>
<tr>
<td>WT</td>
<td>R*4</td>
<td>001444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>$DATA 000376</td>
<td>001002</td>
<td>(257) (257)</td>
</tr>
<tr>
<td>PROMT</td>
<td>L*1</td>
<td>$DATA 001400</td>
<td>000020</td>
<td>(8) (16)</td>
</tr>
<tr>
<td>STRING</td>
<td>L*1</td>
<td>$DATA 0000376</td>
<td>000000</td>
<td>(127) (254)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td>SEND</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE EDITOR

PURPOSE: TO EDIT NUMERICAL CONTROL PROGRAMS ON DISKETTE

INPUT: FILES FROM DISKETTE AND EDITING COMMANDS FROM CRT

PROCESSING: THIS ROUTINE MERELY TRANSFERS CONTROL TO THE OPERATING SYSTEM SO THAT THE SYSTEM EDITOR MAY BE USED TO MODIFY FILES. WHEN EDITING IS COMPLETE CYR02 MUST AGAIN BE CALL FROM THE SYSTEM PROMPT.

OUTPUT: FILES RETURNED TO DISKETTE

CALLED BY: CYR02

CALLS TO: RETURNS TO RT-11

SPECIAL INTERFACE REQUIREMENTS: NONE

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

DATE PROGRAMMER =====CONTACT/ACTION/REMARKS====
11-JUL-85 FRED R. SIAS, JR. (803)-656-3375/NARRATION
3-SEP-85 FRED R. SIAS, JR. MERELY RETURNS TO SYSTEM FOR EDITOR

C
0002 INTEGER RT,WT
0003 INTEGER*2 INPUT
C
0004 WT=7
0005 RT=5
C
0006 WRITE (WT,100)
0007 100 FORMAT(' This call merely returns you to the operating system',/,
1 ' from which you may call the standard editor. After editing',/,
2 ' your numerical control program, again execute CYR02 to',
3 ' continue.'//,
4 ' DO YOU WISH TO EXIT TO THE OPERATING SYSTEM? (Y OR N)> ','$)
C
0008 READ (RT,200) INPUT
0009 200 FORMAT(A2)
C
0010 IF (INPUT.EQ.'Y') STOP
0012 RETURN
0013 END
FORTRAN IV Storage Map for Program Unit EDITOR

Local Variables, .PSECT $DATA; Size = 000006 ( 3. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>000000</td>
</tr>
<tr>
<td>WT</td>
<td>I*2</td>
<td>000002</td>
</tr>
</tbody>
</table>
C**********************************************************************
C
C MODULE NAME:
C
SUBROUTINE PARK
C
C PURPOSE: MENU SELECTION TO 'PARK' THE PROGRAM WHERE
IT WILL IGNORE ALL SPURIOUS MESSAGES FROM ROBOT.
C
C INPUT: ANY KEY PRESS TO EXIT ENDLESS LOOP.
C
C PROCESSING: EXECUTES RECEIVE CONTINUOUSLY IN AN ENDLESS LOOP,
HANDLES ALL COMMUNICATIONS HANDSHAKING,
EXITS WHEN ANY KEY PRESSED.
C
C OUTPUT: MESSAGES TO CRT.
C
C CALLED BY: CYRO2
C
C CALLS TO: RECEIVE
C
C SPECIAL INTERFACE REQUIREMENTS: NONE
C
C REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)
C DATE PROGRAMMER CONTACT/ACTION/REMARKS
C 9-6-85 FRED R. SIAS, JR. (803)-656-3375/PROGRAMMED
C
C**********************************************************************

INTEGER RT,WT
INTEGER K2

WT=7
RT=5

WRITE (WT,1005)
1005 FORMAT(12,'/','Program is now 'Parked' in an endless loop',/,'that will ignore all messages from Robot',/,'PRESS ANY KEY TO RETURN TO MENU.','$/')

CONTINUE

CALL RECEIVE(ERR,IREN,ICSUM,DATA)
IF (ERR.EQ.1) GOTO 9000
GOTO 2000

CONTINUE

RETURN

END
FORTRAN IV Storage Map for Program Unit PARK

Local Variables, .PSECT $DATA, Size = 000016 (7, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICSUM</td>
<td>I*2</td>
<td>000012</td>
</tr>
<tr>
<td>ILEN</td>
<td>I*2</td>
<td>000010</td>
</tr>
<tr>
<td>WT</td>
<td>I*2</td>
<td>000002</td>
</tr>
<tr>
<td>IDATA</td>
<td>I*2</td>
<td>000014</td>
</tr>
<tr>
<td>INPUT</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>IERROR</td>
<td>I*2</td>
<td>000006</td>
</tr>
<tr>
<td>RT</td>
<td>I*2</td>
<td>000000</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>R*4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODULE NAME: SUBROUTINE INTER

PURPOSE: THIS MODULE HANDLES COMMUNICATIONS BETWEEN THE MINC AND THE INTERGRAPH SYSTEM. THE MAIN PURPOSE IS TO DOWN LOAD ROBOT N/C PROGRAMS THAT HAVE BEEN CREATED ON THE INTERGRAPH SYSTEM AND STORE THEM ON THE MINC DISKETTE.

INPUT: INTERACTIVE CONTROL FROM THE KEYBOARD, AND PROGRAM DOWNLOADING FROM INTERGRAPH, INCLUDING HANDSHAKING.

PROCESSING: PROGRAM VERIFICATION.

OUTPUT: STORES DOWNLOADED PROGRAM ON DK:DISKETTE.

CALLED BY: CYRO2

CALLS TO: RETURNS TO RT-11

SPECIAL INTERFACE REQUIREMENTS: SERIAL COMMUNICATIONS PORT.

REVISION HISTORY (REVISE THIS NARRATIVE IF NECESSARY)

<table>
<thead>
<tr>
<th>DATE</th>
<th>PROGRAMMER</th>
<th>CONTACT/ACTION/REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-31-85</td>
<td>FRED R. SIAS, JR.</td>
<td>(803)-656-3375/SYSTEM DESIGN</td>
</tr>
<tr>
<td>1-10-86</td>
<td>DAIVD A. STILES</td>
<td>FINISHED SUBROUTINE RETURNS TO RT-11</td>
</tr>
</tbody>
</table>

INTEGER RT, WT
INTEGER*2 INPUT

WT=7
RT=5

WRITE (WT,100)
FORMAT(' This call merely returns you to the operating system',/,
1 ' from which you may run vtcom.rel to do the downloading of',/,
2 ' your numerical control program, again execute CYRO2 to',
3 ' continue',/,
4 ' DO YOU WISH TO EXIT TO THE OPERATING SYSTEM? (Y OR N)> ','$)

READ (RT,200) INPUT
FORMAT(A2)

IF (INPUT, EQ, 'Y') STOP
RETURN
END
FORTRAN IV Storage Map for Program Unit INTER

Local Variables, PSECT $DATA, Size = 000006 (3, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>I*2</td>
<td>000004</td>
<td>RT</td>
<td>I*2</td>
<td>000000</td>
<td>WT</td>
<td>I*2</td>
<td>000002</td>
</tr>
</tbody>
</table>