FINAL TECHNICAL REPORT

ROBOT WELDING PROCESS CONTROL DEVELOPMENT TASK

9 May 1991 through 8 May 1992
Contract Number NAS8-36955
Delivery Order #120

Prepared for:

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

19 October 1992

by

Peter L. Romine

Electrical and Computer Engineering Department
The University of Alabama in Huntsville
Huntsville, Alabama 35899

(NASA-CR-~~191429~~ 192434) ROBOT WELDING PROCESS CONTROL DEVELOPMENT TASK Final Report, 9 May 1991 - 8 May 1992 (Alabama Univ.) 34 p

N93-18851

Unclas

G3/37 0142858

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Robot Welding Process Control Development Task		5. Report Date Dec 1992 6. Performing Organization Code
7. Author(s) Peter L. Romine		8. Performing Organization Report No. 10. Work Unit No.
9. Performing Organization Name and Address University of Alabama in Huntsville Electrical and Computer Engineering Department Huntsville, AL 35899		11. Contract or Grant No. NAS8-36955 DO #120
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001 George C. Marshall Space Flight Center		13. Type of Report and Period Covered Final Technical May 1992 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The final technical report for the period 9 May 1991 through 8 May 1992.

17. Key Words (Suggested by Author(s))	18. Distribution Statement Unclassified

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	33	

TABLE OF CONTENTS

1.0 INTRODUCTION

This report documents the completion of, and improvements made to, the software developed during 1990 for program maintenance on the PC and HEURIKON and transfer to the CYRO, and integration of the Rocketdyne vision software with the CYRO. The new programs have been used successfully by NASA, Rocketdyne, and UAH technicians and engineers to create, modify, upload, download, and control CYRO NC programs.

2.0 SOFTWARE DEVELOPMENTS

2.1 Introduction

Software development was concentrated in three areas:

- Improve user interface for easy use by persons not experienced in computer operation.
- Simplify the software cycle for new CYRO executive tapes to eliminate the need to travel between three sites with magnetic tapes.
- Modification of Rocketdyne vision software to use the HEURIKON serial port to send offsets to the CYRO through its serial port.

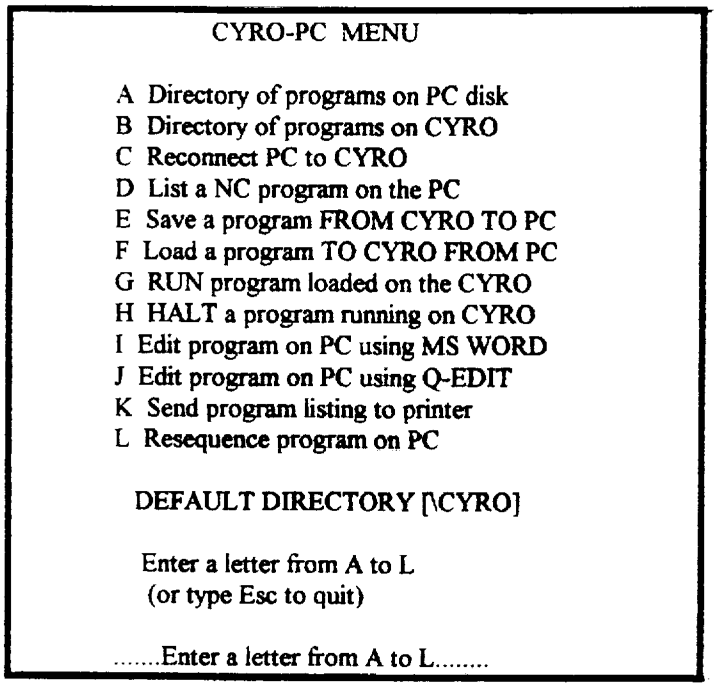
The approach in all software development is to start development on the PC, and carry this development on the PC as long as possible. This is desirable due to the superiority of available editors, compilers, debug tools, and development utilities for the PC that are not available or are very costly on the HEURIKON or similar system. Further, work can be accomplished at any site with a PC; this is especially important considering the harsh working environment of the Building 4705 highbay.

The programs are designed to be easily ported from one machine environment to another. Once a program is operating properly on the PC it is then transferred to the HEURIKON.

2.2 User Interface

The software developed in 1990 required the operator to remember the name and proper usage of six to ten programs. It was quickly discovered, during training sessions, that this was not acceptable, considering the diversity of the users and the potentially long period of time between uses of the software.

A new menu-driven user interface, Figure 1, was developed to go on top of the existing robot communication and control software. The menu is built from the batch processing commands that are a standard part of DOS on the PC and translatable to the shell commands available on the HEURIKON.



```
                CYRO-PC  MENU

A  Directory of programs on PC disk
B  Directory of programs on CYRO
C  Reconnect PC to CYRO
D  List a NC program on the PC
E  Save a program FROM CYRO TO PC
F  Load a program TO CYRO FROM PC
G  RUN program loaded on the CYRO
H  HALT a program running on CYRO
I  Edit program on PC using MS WORD
J  Edit program on PC using Q-EDIT
K  Send program listing to printer
L  Resequence program on PC

   DEFAULT DIRECTORY [\CYRO]

    Enter a letter from A to L
    (or type Esc to quit)

.......Enter a letter from A to L........
```

Figure 1. New user interface menu.

The new user interface is now the preferred way NC programs are maintained and loaded to the CYRO.

2.3 CYRO Tape Development

The previous software development cycle for new CYRO executive tapes allowed the programmer to edit and compile the CYRO program remotely using the network. However, it was necessary to then physically transfer, via magnetic tape, the new executable to a third machine equipped to write the new executive tape on a TU58 data cartridge. This process discouraged development of new executive tapes. New programs were developed to streamline the development cycle.

The problem was solved by developing two programs, BOOTCYRO.EXE and MAKETAPE.EXE. Through research into the structure of the PDP11/23 boot image and TU58 tape drive, the BOOTCYRO program was developed to boot the CYRO directly from a captured boot image file via the PC. With the CYRO serial card set to 38.4 K BAUD, the time required to boot the CYRO was lowered from the 120 seconds required by tape to 20 seconds with the PC.

The initial version of the program required an input file captured from an existing executive tape. Later versions were developed to accept the SEN.TSK file directly as generated by the SCATS compiler. The BOOTCYRO program now allows a programmer to modify the CYRO program, compile and link to create a new .TSK file, transfer the .TSK file over the network to the PC or HEURIKON, and boot the CYRO with the new executive program, all without leaving the CYRO workcell in building 4705.

The MAKETAPE program was developed to support writing of TU58 tapes via the PC or HEURIKON, using the TU58 tape drive built into the CYRO. This program can take the raw .TSK file and create a new executive tape or it can make a copy of an existing TU58 tape. This program also eliminates the need for the RT11UTL, EXCHANGE, and ZAPTU58 programs in the development cycle.

2.4 HEURIKON Vision Software

Software was developed by Rocketdyne in California, to accept images of the weld path gathered by a camera mounted in the torch, perform image processing on the images to detect the seam, and then send offsets to the robot in order to track the seam. The system was developed and demonstrated to work in California on a similar HEURIKON computer system with a different robot. It is desired to adapt this software to operate on the HEURIKON computer at MSFC with the CYRO robot in building 4705.

In discussions with Dave Gutow of Rocketdyne Canoga Park, it was determined that the special communications between the vision software on the HEURIKON and the CYRO is best handled in the SENCON.C program file. Dave Gutow designed his software to use a shared data structure and the functions in SENCON.C were written with provision for the requirements at MSFC.

In Canoga Park, a seperate processor card communicates with the robot and updates the shared data structure. At MSFC, a software process running under UNIX has been developed to communicate with the CYRO and alter the data structure as appropriate.

3.0 HARDWARE DEVELOPMENTS

3.1 CYRO Air-conditioning

The extreme heat generated in the HEURIKON cabinet, worsened by the high temperatures experienced in the un air-conditioned CYRO work cell, resulted in a disk crash in July 1991. Several months were required to rebuild the system. A dedicated air-conditioner was installed on the HEURIKON cabinet to prevent future crashes and prolong the life of the HEURIKONs components.

3.2 CYRO Tape Switch

A rotary switch was added to the CYRO front panel to simplify the connections between the PC/HEURIKON, CYRO TU58, and CYRO Serial Port. The knob has three positions to allow the CYRO to boot from tape, boot from an external device, or connect external device to the TU58.

4.0 WELDING PROCESS MEASUREMENT AND CONTROL SYSTEMS

4.1 MIDSOUTH Model-Based System

The MIDSOUTH was delivered and demonstrated. The system contains an 80286 processor board, 80386 processor board, data acquisition, and signal conditioning hardware in a VME cardcage. The process monitoring and control software is written in C to operate under Microsoft windows.

Basic operation of the system was verified to assist in evaluation of the possible applications of the system, as-is or as a parts platform for other projects.

4.2 INTA Laser Seam Tracker

The INTA laser seam tracker was delivered in pieces, non-operational, and with limited documentation. Inaccuracies in the wiring diagrams were discovered during the re-connection of the units.

The MIDSOUTH system was used to download the INTA software to the 68030 processor; this requires 2 to 3 minutes. A BIT3 interface card was included but requires a 16-bit ISA slot. The BIT3 card is claimed to complete a download in 15 seconds.

The INTA system detects the seam by measuring scatter laser light. The light is applied to the work-piece by reflecting a laser beam emitted from a fiber optic cable. A semiconductor laser is connected to a lens/collimator assembly by the fiber optic cable. The lens assembly includes a pinhole which must be precisely aligned to allow the laser light to pass through. This alignment was disturbed in shipping or during assembly and several weeks were required to realign it.

After realignment, the system was checked-out and it operated as described in the documentation.

5.0 CONCLUSIONS AND RECOMMENDATIONS

The PC version of the new NC development software has been used by most of the CYRO users. The HEURIKON version has not been warmly received due to the greater complexity in operating the HEURIKON and the unfamiliar vi editor.

The arrival of the new robots and welding controllers has prompted a reevaluation of the focus and role of the CYRO and HEURIKON in the NASA mission. The welding controller originally planned for the CYRO has now been teamed with a more modern robot arm. The serial communications interface of the CYRO and HEURIKON is now very robust and well understood. Electrically, the CYRO and/or HEURIKON can now be easily connected to any of the new robot or welding controllers. The age of the HEURIKON technology is now an important consideration in its role with welding process control. However, the VME-based construction of the HEURIKON will support upgrades to faster processors and peripherals, well into the future.

The vision system changes are still to be tested. A compatibility problem with the compiler and configuration and/or hardware problems with the VRTX processor have hampered this development. Progress with the vision system was severly hampered due to the lack of modern software development tools.

The HEURIKON, MIDSOUTH, and INTA a three separate systems that have never been fully functional. The HEURIKON system software is out of date and lacking. Both the HEURIKON and INTA have powerful and reusable hardware. The INTA system does not have a disk or operating system. Combining the HEURIKON and INTA hardware and adding modern, up to date software tools would make for a very powerful system for welding process control.

The PC nature of the MIDSOUTH system makes for a powerful software development platform. However, the VME based hardware has not proven ideal combination to the point.

Due to the relatively slow rates encountered in the welding process, the serial interfaces common to the CYRO, HEURIKON, MIDSOUTH, INTA, and new robot, are sufficient. As the need arises for higher speed communications, high speed interface cards such as those made by BIT3 will work.

APPENDICES

APPENDIX A

CYRO BOOT and TU58 Software Listings

```
/* BOOTCYRO.C  --- Program to boot the CYRO750 ROBOT from a .TSK image file*/
/* Written by       Peter L. Romine   1990,91
**                  University of Alabama, Huntsville
**                  Electrical and Computer Engineering Department
*/
#include "cyro.h"
#pragma    check_stack (off)
static CYRO    msg;
UCHAR
blk0[]={176,0,225,121,0,0,254,118,192,101,30,0,1,148,3,3,247,9,6,0,251,1,5,0,0,0,95,144
,118,255,223,139,116,255,253,128,135,0};
UCHAR blk6[]={1,0,0,0,1,0,0,0,8,0,0,4,225,121,0,0,254,118,110,0,0,0,179,7,0,2,0,
0,0,0,0,0,138,1,0,0,0,0,0,8};
#define DATASIZ 60000
static UCHAR   data[DATASIZ];
long    data_ptr,data_len;
int     pkt=0;
struct
{
        UCHAR   flag;
        UCHAR   len;
        UCHAR   opcode;
        UCHAR   modifier;
        UCHAR   unit;
        UCHAR   switches;
        UCHAR   seq_lo;
        UCHAR   seq_hi;
        UCHAR   cnt_lo;
        UCHAR   cnt_hi;
        UCHAR   blk_lo;
        UCHAR   blk_hi;
        UCHAR   chk_lo;
        UCHAR   chk_hi;
} cmd_pkt;
struct
{
        UCHAR   flag;
        UCHAR   len;
        UCHAR   data[128];
        UCHAR   chk_lo;
        UCHAR   chk_hi;
} data_pkt;

int main(int argc,char **argv)
{
        int     blk_strt,blk_end;
        FILE    *fp,*fp0;
        long    byte_cnt,i;
        UCHAR   chk_l,chk_h;
        long    lobyte,hibyte;
        long    chk,a=0,b=0,c=0;

        connect_serial(COMM2,RESTORE_ON_EXIT);
        if( !(fp = fopen(argv[1],"rb")) )
                exit(1);

        for(i=0; i<DATASIZ; i++) data[i] = 0;
        printf("Rd 0  %d\n",a=fread(data,1,(size_t)1024,fp));
        fp0 = fopen("tu58_4.cln","rb");
        printf("Rd 1  %d\n",a=fread(data,1,(size_t)2048,fp0));
        printf("Rd 2  %d\n",b=fread(&(data[2048]),1,(size_t)30720,fp)); /* rd blk 0 */
        printf("Rd 3  %d\n",c=fread(&(data[32768]),1,(size_t)30720,fp)); /* rd blk 8 */
        fclose(fp0);

        data_ptr = 0L;
        data_len = a + b + c;
        printf("BUFFER SIZE = %ld\n",data_len);
        printf("\nWaiting for INIT from CYRO\n"); while( s_getch2() != 0x04 );
        while( s_getch2() != 0x04 );
        s_putch2(0x10); /* send continue */
        /* get command to read block 0 */
        get_command();
        print_command();
        printf("\nSending block 0 to CYRO\n"); /* Send block 0 */
        send_data();
        /* get command to read block 6 */
        get_command();
        print_command();
        printf("\nSending dir to CYRO\n"); /* Send block 6 */
        send_data();
        /* get command to read block 8 */
        get_command();
        print_command();
        printf("\nSending Program to CYRO\n");
        /* Send block 8a */
        send_data();
        printf("\nTransmission to CYRO complete!\n");
}

print_command()
{
        printf("%d %d %d %d %d %d %d %d %d %d %d %d %d\n", cmd_pkt.flag
                ,cmd_pkt.len
                ,cmd_pkt.opcode
                ,cmd_pkt.modifier
                ,cmd_pkt.unit
                ,cmd_pkt.switches
                ,cmd_pkt.seq_lo
                ,cmd_pkt.seq_hi
                ,cmd_pkt.cnt_lo
                ,cmd_pkt.cnt_hi
                ,cmd_pkt.blk_lo
                ,cmd_pkt.blk_hi
                ,cmd_pkt.chk_lo
                ,cmd_pkt.chk_hi);
}

get_command()
{
        cmd_pkt.flag            = s_getch2();
        cmd_pkt.len                     = s_getch2();
        cmd_pkt.opcode          = s_getch2();
        cmd_pkt.modifier        = s_getch2();
        cmd_pkt.unit            = s_getch2();
        cmd_pkt.switches        = s_getch2();
        cmd_pkt.seq_lo          = s_getch2();
        cmd_pkt.seq_hi          = s_getch2();
        cmd_pkt.cnt_lo          = s_getch2();
        cmd_pkt.cnt_hi          = s_getch2();
        cmd_pkt.blk_lo          = s_getch2();
        cmd_pkt.blk_hi          = s_getch2();
        cmd_pkt.chk_lo          = s_getch2();
        cmd_pkt.chk_hi          = s_getch2();
}

send_data()
{
        long    blk_strt,blk_end,i;
        long    byte_cnt;
        UCHAR   chk_l,chk_h;
        long    lobyte,hibyte;
        unsigned long  chk;

        byte_cnt = 256L*(long)(cmd_pkt.cnt_hi) + (long)(cmd_pkt.cnt_lo);
        while( byte_cnt )
        {
                s_putch2(1);
                s_putch2(128);
                chk = 128L*256L + 1L;
                blk_strt = data_ptr;
                blk_end = blk_strt + 128L;
                for(i=blk_strt; i<blk_end; i+=2L)
                {
                        s_putch2(data[i]);
                        s_putch2(data[i+1]);
                        chk += (unsigned long)data[i]
                        + (unsigned long)(256L*(unsigned long)data[i+1]);
                        if( chk > 0xffffL )
                                chk -= 0xffffL;
                }
                chk_h = (UCHAR)(chk/256L);
                chk_l = (UCHAR)(chk - ((unsigned long)chk_h * 256L));
                s_putch2(chk_l);
                s_putch2(chk_h);
                byte_cnt -= 128L;
                data_ptr += 128L;
                pkt++;
                if( !(pkt%10) )
                        printf("Packet %d\r",pkt);
        }
        chk = 2626L;
        chk += (unsigned long)(cmd_pkt.cnt_lo)
        + (unsigned long)(256L*(unsigned long)(cmd_pkt.cnt_hi));
        if( chk > 0xffffL )
                chk -= 0xffffL;
        chk_h = (UCHAR)(chk/256L);
        chk_l = (UCHAR)(chk - ((unsigned long)chk_h * 256L));
        s_putch2(2);
        s_putch2(10);
        s_putch2(64);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(cmd_pkt.cnt_lo);
        s_putch2(cmd_pkt.cnt_hi);
        s_putch2(0);
        s_putch2(0);
        s_putch2(chk_l);
        s_putch2(chk_h);
}
```

```
/* CAPTURE.C --- Program to capture the data loaded into the CYRO750 ROBOT during */
/*                    a boot from tape*/

/* Written by            Peter L. Romine  1990,91
**                       University of Alabama, Huntsville
**                       Electrical and Computer Engineering Department
*/

#include "cyro.h"

#pragma    check_stack (off)

FILE     *fp;

int s_getc2(void)
{
        while( !inp_cnt2() )
        if( kbhit() )
        {
                fclose(fp);
                exit(1);
        }

        return inp_char2();
}


int main(int argc,char **argv)
{
        connect_serial(COMM2,RESTORE_ON_EXIT);

        if( !(fp = fopen(argv[1],"wb")) )
        {
                perror(argv[1]);
                exit(1);
        }

        while( 1 )
                fputc(s_getc2(),fp);
}
```

~~PRECEDING PAGE BLANK NOT FILMED~~

```
/* MAKETAPE.C --- Program to create a new executive tape using the TU58 drive */
                   built into the CYRO750 ROBOT */

/* Written by          Peter L. Romine 1990,91
**                     University of Alabama, Huntsville
**                     Electrical and Computer Engineering Department
*/

#include "cyro.h"

#pragma   check_stack (off)

static CYRO       msg;

#define  DATASIZ        60000

static UCHAR     data[DATASIZ];
long       data_ptr,data_len;
int        pkt=0;
struct
{
           UCHAR flag;
           UCHAR len;
           UCHAR opcode;
           UCHAR modifier;
           UCHAR unit;
           UCHAR switches;
           UCHAR seq_lo;
           UCHAR seq_hi;
           UCHAR cnt_lo;
           UCHAR cnt_hi;
           UCHAR blk_lo;
           UCHAR blk_hi;
           UCHAR chk_lo;
           UCHAR chk_hi;
} cmd_pkt;

struct
{
           UCHAR flag;
           UCHAR len;
           UCHAR data[128];
           UCHAR chk_lo;
           UCHAR chk_hi;
} data_pkt;
```

```
int main(int argc,char **argv)
{
        int             blk_strt,blk_end,ch;
        FILE    *fp,*fp0;
        long    byte_cnt,i;
        UCHAR chk_l,chk_h,s;
        long    lobyte,hibyte;
        long    chk,a=0,b=0,c=0;
        UCHAR cnt_l,cnt_h;

        connect_serial(COMM2,RESTORE_ON_EXIT);

        if( !(fp = fopen(argv[1],"rb")) )
                exit(1);

        /* Initialize the data array to all 0's */
        for(i=0; i<DATASIZ; i++)
                data[i] = 0;

        /* Skip over 1st 1024 bytes of .tsk image */
        printf("Rd 0  %d\n",a=fread(data,1,(size_t)1024,fp));

        /* Read 1st 2048 bytes from a .cln file */
        fp0 = fopen("tu58_4.cln","rb");
        printf("Rd %d\n",a=fread(data,1,(size_t)2048,fp0));
        fclose(fp0);

        /* Read the program from the .tsk file */
        printf("Rd %d\n",b=fread(&(data[2048]),1,(size_t)30720,fp)); /* rd prt 1 */
        printf("Rd %d\n",c=fread(&(data[32768]),1,(size_t)30720,fp));/* rd prt 2 */
        fclose(fp);

        data_ptr = 0L;
        data_len = a + b + c;
        printf("BUFFER SIZE = %ld\n",data_len);

        /* Send a break to the TU58
        printf("Sending BREAKS to TU58\n");
        send_break2();
        send_break2();
        send_break2();
        */

        printf("Sending INITs to TU58\n");
        s_putch2(0x04);
        s_putch2(0x04);
        /* wait for CONTINUE from TU58 */
        while( (ch=s_getch2()) != 0x10 )
                printf("%d ",ch);
```

```
/* Command Packet to write Block 0 */
cmd_pkt.flag      = 0x02;
cmd_pkt.len       = 0x0a;
cmd_pkt.opcode    = 0x03;
cmd_pkt.modifier  = 0x00;
cmd_pkt.unit      = 0x00;
cmd_pkt.switches  = 0x00;
cmd_pkt.seq_lo    = 0x00;
cmd_pkt.seq_hi    = 0x00;
cmd_pkt.cnt_lo    = 0x00;
cmd_pkt.cnt_hi    = 0x02;
cmd_pkt.blk_lo    = 0x00;
cmd_pkt.blk_hi    = 0x00;

send_command();

printf("\nWriting BLOCK 0 to TU58\n");
write_data(512L);
get_command();
printf("OPCODE       = %d\n",(int)cmd_pkt.opcode);
printf("Success Code = %d\n",(int)cmd_pkt.modifier);
printf("BYTE Count   = %ld\n",256L*(long)(cmd_pkt.cnt_hi) + (long)(cmd_pkt.cnt_lo));
printf("STATUS       = %d %d\n",(int)cmd_pkt.blk_lo,(int)cmd_pkt.blk_hi);

cmd_pkt.flag      = 0x02;
cmd_pkt.len       = 0x0a;
cmd_pkt.opcode    = 0x03;
cmd_pkt.modifier  = 0x00;
cmd_pkt.unit      = 0x00;
cmd_pkt.switches  = 0x00;
cmd_pkt.seq_lo    = 0x00;
cmd_pkt.seq_hi    = 0x00;
cmd_pkt.cnt_lo    = 0x00;
cmd_pkt.cnt_hi    = 0x04;
cmd_pkt.blk_lo    = 0x06;
cmd_pkt.blk_hi    = 0x00;

send_command();

printf("\nWriting BLOCK 6 to TU58\n");
write_data(1024L);
get_command();
printf("Success Code = %d\n",(int)cmd_pkt.modifier);
printf("BYTE Count   = %ld\n",256L*(long)(cmd_pkt.cnt_hi) + (long)(cmd_pkt.cnt_lo));
printf("STATUS       = %d %d\n",(int)cmd_pkt.blk_lo,(int)cmd_pkt.blk_hi);

/* send WRITE command */
data_len -= 1536L;
cnt_h = (UCHAR)(data_len/256L);
cnt_l = (UCHAR)(data_len - ((unsigned long)cnt_h * 256L));

cmd_pkt.flag      = 0x02;
cmd_pkt.len       = 0x0a;
cmd_pkt.opcode    = 0x03;
```

```
        cmd_pkt.modifier  = 0x00;
        cmd_pkt.unit      = 0x00;
        cmd_pkt.switches  = 0x00;
        cmd_pkt.seq_lo    = 0x00;
        cmd_pkt.seq_hi    = 0x00;
        cmd_pkt.cnt_lo    = cnt_l;
        cmd_pkt.cnt_hi    = cnt_h;
        cmd_pkt.blk_lo    = 0x08;
        cmd_pkt.blk_hi    = 0x00;

        send_command();

        printf("\nWriting BLOCK 8 to TU58\n");
        write_data(data_len);
        get_command();
        printf("Success Code = %d\n",(int)cmd_pkt.modifier);
        printf("BYTE Count   = %ld\n",256L*(long)(cmd_pkt.cnt_hi) + (long)(cmd_pkt.cnt_lo));
        printf("STATUS       = %d %d\n",(int)cmd_pkt.blk_lo,(int)cmd_pkt.blk_hi);

        printf("\nTransmission to TU58 complete!\n");
}

get_command()
{
        cmd_pkt.flag            = s_getch2();
        cmd_pkt.len                     = s_getch2();
        cmd_pkt.opcode          = s_getch2();
        cmd_pkt.modifier= s_getch2();
        cmd_pkt.unit            = s_getch2();
        cmd_pkt.switches= s_getch2();
        cmd_pkt.seq_lo          = s_getch2();
        cmd_pkt.seq_hi          = s_getch2();
        cmd_pkt.cnt_lo          = s_getch2();
        cmd_pkt.cnt_hi          = s_getch2();
        cmd_pkt.blk_lo          = s_getch2();
        cmd_pkt.blk_hi          = s_getch2();
        cmd_pkt.chk_lo          = s_getch2();
        cmd_pkt.chk_hi          = s_getch2();
}


send_command()
{
        unsigned long     chk;
        UCHAR chk_l,chk_h;

        chk = (unsigned long)(cmd_pkt.len)*256L + (unsigned long)cmd_pkt.flag;
        chk += (unsigned long)(cmd_pkt.modifier)*256L + (unsigned long)cmd_pkt.opcode;
        chk += (unsigned long)(cmd_pkt.switches)*256L + (unsigned long)cmd_pkt.unit;
        chk += (unsigned long)(cmd_pkt.seq_hi)*256L + (unsigned long)cmd_pkt.seq_lo;
        chk += (unsigned long)(cmd_pkt.cnt_hi)*256L + (unsigned long)cmd_pkt.cnt_lo;
        chk += (unsigned long)(cmd_pkt.blk_hi)*256L + (unsigned long)cmd_pkt.blk_lo;
        chk_h = (UCHAR)(chk/256L);
        chk_l = (UCHAR)(chk - ((unsigned long)chk_h * 256L));
        s_putch2(cmd_pkt.flag);
        s_putch2(cmd_pkt.len);
        s_putch2(cmd_pkt.opcode);
        s_putch2(cmd_pkt.modifier);
        s_putch2(cmd_pkt.unit);
        s_putch2(cmd_pkt.switches);
        s_putch2(cmd_pkt.seq_lo);
        s_putch2(cmd_pkt.seq_hi);
        s_putch2(cmd_pkt.cnt_lo);
        s_putch2(cmd_pkt.cnt_hi);
        s_putch2(cmd_pkt.blk_lo);
        s_putch2(cmd_pkt.blk_hi);
        s_putch2(chk_l);
        s_putch2(chk_h);
}

write_data(byte_cnti)
        long    byte_cnti;
{
        long    byte_cnt,blk_strt,blk_end,i;
        UCHAR chk_l,chk_h,cnt_h,cnt_l,c;
        long    lobyte,hibyte;
        unsigned long   chk;

        byte_cnt = byte_cnti;

        while( byte_cnt )
        {
                /* wait for continue */
                while( (c=s_getch2()) != 0x10 )
                        putchar(c);

                s_putch2(1);
                s_putch2(128);

                chk = 128L*256L + 1L;

                blk_strt = data_ptr;
                blk_end = blk_strt + 128L;

                for(i=blk_strt; i<blk_end; i+=2L)
                {
                        s_putch2(data[i]);
                        s_putch2(data[i+1]);

                        chk += (unsigned long)data[i]
                         + (unsigned long)(256L*(unsigned long)data[i+1]);
                        if( chk > 0xffffL )
                                chk -= 0xffffL;
                }

                chk_h = (UCHAR)(chk/256L);
                chk_l = (UCHAR)(chk - ((unsigned long)chk_h * 256L));
```

```
		s_putch2(chk_l);
		s_putch2(chk_h);

		byte_cnt -= 128L;

		data_ptr += 128L;

		pkt++;
		if( !(pkt%10) )
			printf("Packet %d\r",pkt);
	}

	return;
}
```

```
/* READTAPE.C -- Program to read an executive tape from the  CYRO750 ROBOT */

/* Written by            Peter L. Romine   1990,91
**                       University of Alabama, Huntsville
**                       Electrical and Computer Engineering Department
*/

#include "cyro.h"

#pragma   check_stack (off)

static CYRO      msg;

#define  DATASIZ         60000

static UCHAR     data[DATASIZ];
long       data_ptr,data_len;
int        pkt=0;
struct
{
           UCHAR flag;
           UCHAR len;
           UCHAR opcode:
           UCHAR modifier;
           UCHAR unit;
           UCHAR switches;
           UCHAR seq_lo;
           UCHAR seq_hi;
           UCHAR cnt_lo;
           UCHAR cnt_hi;
           UCHAR blk_lo;
           UCHAR blk_hi;
           UCHAR chk_lo;
           UCHAR chk_hi;
} cmd_pkt;

struct
{
           UCHAR flag;
           UCHAR len;
           UCHAR data[128];
           UCHAR chk_lo;
           UCHAR chk_hi;
} data_pkt;
```

```
int main(int argc,char **argv)
{
        int                  blk_strt,blk_end;
        FILE     *fp,*fp0;
        long     byte_cnt,i;
        UCHAR chk_l,chk_h,s;
        long     lobyte,hibyte;
        long     chk,a=0,b=0,c=0;
        UCHAR cnt_l,cnt_h;

        connect_serial(COMM2,RESTORE_ON_EXIT);

        /* Send command string to read the entire tape contents */

        s_putch2(2);
        s_putch2(10);
        s_putch2(2);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(0);
        s_putch2(2);
        s_putch2(0);
        s_putch2(0);
        s_putch2(4);
        s_putch2(12);

        while(1)
                printf("%d ",s_getch2());

        exit(1);
}
```

```
/* RESEQ.C  — Program to resequence a NC program for the CYRO750 ROBOT */
/* Written by              Peter L. Romine   1990,91
**                         University of Alabama, Huntsville
**                         Electrical and Computer Engineering Department
*/

#include <cyro.h>

static char str1[100],str2[100];

main(argc,argv)
        int     argc;
        char    **argv;
{
        FILE    *fpin,*fpout;
        char    str[80],fname[12],numstr[7];
        int             i,j,num,len;

        if( argc > 2 )
        {
                printf("\nUSAGE: reseqp file.nc\n");
                exit(1);
        }

        if( argc == 2 )
                strcpy(fname,argv[1]);
        else
        {
                printf("\nWHAT PROGRAM NAME ? ");
                scanf("%s",fname);
                fflush(stdin);
        }

        if( !(fpin=fopen(fname,"r")) )
        {
                perror(fname);
                exit(1);
        }

        strcpy(str1,"copy ");
        strcat(str1,fname);
        strcat(str1," ");
        strcat(str1,fname);
        str1[strlen(str1)-3] = 0;
        strcat(str1,".BAK");
        system(str1);

        if( !(fpout=fopen("RESEQ$$$.TMP","w")) )
        {
                perror("TMPFILE");
                exit(1);
        }

        num = 10;
        while( fgets(str1,90,fpin) )
        {
                sprintf(str,"%d",num);
                if( num < 100 )
                        strcpy(numstr,"N00");
                else if( num < 1000 )
                        strcpy(numstr,"N0");
                else
                        strcpy(numstr,"N");
                strcat(numstr,str);

                strcpy(str2,numstr);

                len = strlen(str1);

                i = 0;
                while( i <= len )
                {
                        switch( toupper(str1[i]) )
                        {
                        case '\r':
                        case '\n':
                        case ';':
                        case 'F':
                        case 'G':
                        case 'L':
                        case 'M':
                        case 'W':
                        case 'V':
//                      case 'K':
                                        strcat(str2,&str1[i]);
                                        i = len + 1;
                                        break;

                        default:
                                i++;
                                break;
                        }
                }

                fputs(str2,fpout);

                num += 10;
        }

        fclose(fpin);
        fclose(fpout);

        strcpy(str1,"copy RESEQ$$$.TMP ");
        strcat(str1,fname);
        system(str1);
}
```

APPENDIX B

New PC Menu Listing

```
@ECHO OFF
REM  CYROMENU.BAT  --- DOS batch file to create user menu.
REM  Written by          Peter L. Romine   1990,91
REM                      University of Alabama, Huntsville
REM                      Electrical and Computer Engineering Department
REM
FMARK MENU >NUL
CALL LOADHLP
:TOP
sa bright white on blue
cls
ECho          +------------------------------------+
ECho          ¦                                    ¦
ECho          ¦       CYRO-PC  MENU                ¦
ECho          ¦                                    ¦
ECho          ¦ A  Directory of programs on PC disk ¦
ECho          ¦ B  Directory of programs on CYRO   ¦
ECho          ¦ C  Reconnect PC to CYRO            ¦
ECho          ¦ D  List a NC program on the PC     ¦
ECho          ¦ E  Save a program FROM CYRO TO PC  ¦
ECho          ¦ F  Load a program TO CYRO FROM PC  ¦
ECho          ¦ G  RUN program loaded on the CYRO  ¦
ECho          ¦ H  HALT a program running on CYRO  ¦
ECho          ¦ I  Edit program on PC using MS WORD ¦
ECho          ¦ J  Edit program on PC using Q-EDIT ¦
ECho          ¦ K  Send program listing to printer ¦
ECho          ¦ L  Resequence program on PC        ¦
ECho          ¦                                    ¦
ECho          ¦                                    ¦
ECho          ¦   DEFAULT DIRECTORY [\CYRO]        ¦
ECho          ¦                                    ¦
ECho          ¦    Enter a letter from A to L      ¦
ECho          ¦     (or type Esc to quit)          ¦
ECho          +------------------------------------+

:START
tick  Enter a letter from A to L
echo .
GETLETR
IF ERRORLEVEL 27 GOTO END
IF ERRORLEVEL 13 GOTO START
IF ERRORLEVEL 12 GOTO LABELL
IF ERRORLEVEL 11 GOTO LABELK
IF ERRORLEVEL 10 GOTO LABELJ
IF ERRORLEVEL 9 GOTO LABELI
IF ERRORLEVEL 8 GOTO LABELH
IF ERRORLEVEL 7 GOTO LABELG
IF ERRORLEVEL 6 GOTO LABELF
IF ERRORLEVEL 5 GOTO LABELE
IF ERRORLEVEL 4 GOTO LABELD
IF ERRORLEVEL 3 GOTO LABELC
IF ERRORLEVEL 2 GOTO LABELB
:LABELA
dir/p *.nc
```

```
ticker
GOTO TOP
:LABELB
dirp
ticker
GOTO TOP
:LABELC
init
ticker
GOTO TOP
:LABELD
listp
ticker
GOTO TOP
:LABELE
savep m
ticker
GOTO TOP
:LABELF
loadp m
ticker
GOTO TOP
:LABELG
runp m
ticker
GOTO TOP
:LABELH
haltp m
ticker
GOTO TOP
:LABELI
echo    Please Wait while WORD is loaded ...
word
GOTO TOP
:LABELJ
q
GOTO TOP
:LABELK
printp
GOTO TOP
:LABELL
reseqp
GOTO TOP
:END
RELEASE MENU >NUL
```

APPENDIX C

HEURIKON Software Listings

```
/* SENCON.C module - This module acts as the SENsor CONtroller interface*/
/* for the T3V system.  Throughout this module there are IFDEF compile  */
/* switches checking for the switch SEPSC.  If this switch is set it    */
/* indicates the presence of a seperate sensor controller board as in   */
/* Rocketdyne configuration.  If this switch is not set it indicates no  */
/* seperate sensor controller board, as in the MSFC configuration.      */
/* The routines included are:                                            */
/*  se_init - initializes the sensor controller and/or communications   */
/*  se_delta - sends deltas to the the sensor controller or robot          */
/*  se_getsystem - gets all system paramaters from the sensor controller*/
/*  se_settime - sets the time (if time kept in SENCON module)             */
/*  se_post - posts a message to the sensor controller                     */

#include "/usr/system/codes.h"
#include "/usr/system/shared.h"
#include "util.h"
#include "system.h"
#include "global.h"

extern STATE state;        /* current system states                  */
extern SYSTEM system;      /* The current system parameters          */
/*************************************************************************/
/* SE_INIT - initializes the sensor controller system.  If a seperate   */
/* board is present it initializes communications with it.  If no         */
/* seperate board it starts up the clock and initializes the on board   */
/* sensor controller system.                                               */

se_init (se_qid)
int se_qid;             /* Queue ID for messages from sensor controller */
{
#ifdef SEPSC            /* Initialize communication if seperate board   */
 vt_msg ("Initializing Intercard Comm routines...\n", state.debugl);
 initcom (se_qid);
#else                   /* Initialize communication if same board     */
/* Init for MSFC */
 init_serial();         /* Initialize the serial console port */
 init_cyro();           /* Wait for the CYRO init msg */
#endif
}

/*************************************************************************/
/* SE_DELTA - sends a cross seam delta to the robot.  Info is sent only */
/* if the control switch is set and the sensor controller data read     */
/* switch is set.  Two parameters are sent; delta and conf.  Delta is   */
/* the cross seam error passed as a velocity in units of .001"/sec.     */
/* Conf is the calculated confidence in the delta (0 - 100).            */

se_delta (delta,  conf)
int delta;           /* The cross seam error, given as a velocity     */
int conf;            /* The confidence in the delta    */
{
 /* Send info only if in control and Sensor Controller data is readable */
 if (state.control && state.scread)
 {
#ifdef SEPSC           /* if we have a sensor controller board...    */
    T3V_FDBCK->delta = delta;
    T3V_FDBCK->confidence = conf;
    se_post (T3VDELTA);
#else                  /* if we don't have a seperate board...       */
/* For MSFC, send an override to the CYRO */
    se_post(delta);
#endif
 }
}
```

```
/**************************************************************************/
/* SE_GETSYSTEM - gets all system data from the sensor controller and   */
/* stores it into the struct SYSTEM.  All data is moved at once to        */
/* maintain consistancy and to (in one place) be able to shut off reads */
/* to the sensor controller.  Data is only read if the sensor controller*/
/* read switch is set.                                                   */

se_getsystem ()
{
#ifdef SEPSC               /* If seperate board...    */
 if (state.scread)
 {          /* OK to read from Sensor controller      */
    system.weldstate = ACTUAL->weldstate;
    system.pulsing   = PROGRMED->pulseonoff;
    system.speed     = PROGRMED->travel;
    system.peakcurr  = ACTUAL->p_cur;
    system.backcurr  = ACTUAL->b_cur;
    system.hours     = TIME->hours;
    system.minutes   = TIME->mins;
    system.secs      = TIME->secs;
 }
 else
 {                  /* not OK to read, set values to default */
    system.weldstate = WELDOFF;
    system.pulsing   = FALSE;
    system.speed     = 0;
    system.peakcurr  = 0;
    system.backcurr  = 0;
    system.hours     = 0;
    system.minutes   = 0;
    system.secs      = 0;
 }
#else               /* If no seperate board   */
/* For MSFC CYRO */
 if (!cyro_locked)
 {          /* OK to read from CYRO structure */
    system.weldstate = cyro.weldstate;
    system.pulsing   = cyro.pulseonoff;
    system.speed     = cyro.travel;
    system.peakcurr  = cyro.p_cur;
    system.backcurr  = cyro.b_cur;
    system.hours     = cyro.hours;
    system.minutes   = cyro.mins;
    system.secs      = cyro.secs;
 }
 else
 {                  /* not OK to read, set values to default */
    system.weldstate = WELDOFF;
    system.pulsing   = FALSE;
    system.speed     = 0;
    system.peakcurr  = 0;
    system.backcurr  = 0;
    system.hours     = 0;
    system.minutes   = 0;
    system.secs      = 0;
 }
#endif
}

/**************************************************************************/
/* SE_POST - posts a message to the sensor controller.  If the sensor     */
/* controller is a seperate board it sends the message via the intercard*/
/* communication system.  The message is sent only if the sensor          */
/* controller read switch is set.                                         */

se_post (msg)
int msg;                    /* The message to post    */
{
#ifdef SEPSC                /* If seperate board      */
 if (state.scread)          /* Send only if SC Read switch set.*/
    post_to_sc (msg);
#else                       /* If no seperate board   */
/* For MSFC, send override msg to CYRO */
    cyro_ovride(msg);
#endif
}
```