```
IIS
┌────────────────────┐
│ F1=SETUP MODE      │
│ F2=PROGRAM MODE    │
│ F3=TEST MODE       │
│ F4=RUN MODE        │
└────────────────────┘

F1  F2  F3  F4
F5  F6  F7  F8
BACK  1  2  3
SPACE
NEXT  4  5  6
PREV  7  8  9
─    •  0  ENTER
```

OPI-7X OPERATOR INTERFACE UNIT

INSTRUCTION BOOK

INDUSTRIAL INDEXING SYSTEMS, Inc.

Revision - 0
Approved By:

ER-6029

ERRATA SHEET, IB-11B024   MARCH 1999

Date	Rev.	ECN No.	DR	CHK	CHK
08/05/97	0	ECN-97-247 (See Note 1)	KY	CD	
03/02/99	A	ECN-99-070 (See Note 2)	KY	CD	KW

Notes:

1)   Cover page dated August 1997 supersedes Cover page dated June 1997. Pages 3 and 12, dated August 1997 supersedes pages 3 and 12, dated June 1997.

2)   Page 30, dated March 1999 supersedes page 30, dated June 1997.

INDUSTRIAL INDEXING SYSTEMS, Inc.

626 Fishers Run
Victor, New York 14564

Tel: (585) 924-9181   Fax: (585) 924-2169

TABLE OF CONTENTS

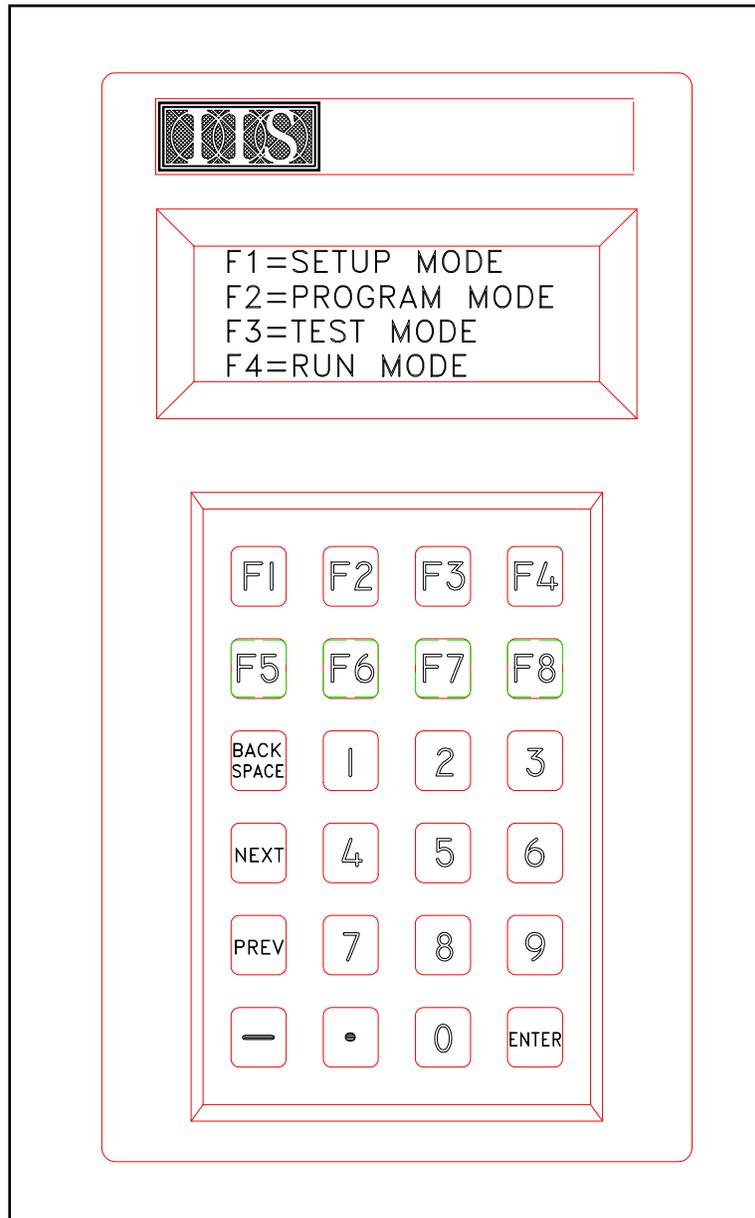## Introduction and Summary of Features

Figure 1.1 - OPI-7X

The OPI-7X is an easy to use and easy to program operator interface device consisting of a keypad and 16 character by 4 line display. The unit has a built-in command code set which allow the user to program menus, program messages, and easily control the cursor and display.

This manual is divided into three sections. Section 1 provides information about the hardware including specifications, mounting dimensions, power-on setup, default configuration and interconnection drawings. Section 2 is for the MSC family of programmable controllers. It details methods of writing Macroprograms to control the display providing the user with several examples and a sample Macroprogram with many useful subroutines. Section 3 contains three appendices; an ASCII Conversion Table, Terminal Code Commands and a Command Code Summary Chart.

Table 1.1 summarizes the features of the OPI operator interface terminal:

○ Full function key menu selection.

○ Full numeric data entry editing.

○ Custom user menu configurations possible with the MSC-Toolkit or MacroPro™ development system software.

○ Industrial panel mount.

○ 115 volt AC power input, no external power supplies needed.

○ Current loop serial interface for ground loop isolation.

○ LCD-backlight display for easy viewing.

Table 1.1

Section 1 - OPI-7X Hardware

1.1   OPI-7X Specifications

Note: All voltages are with respect to the ground line, and all temperatures are in degrees
Celsius.

Supply voltage	115 VAC .1 Amp
Operating temperature range	0 to 50° C
Non-condensing humidity range, storage and operating	0 to 90%

3.98 [101.0]

7.60 [193.0]

Figure 1.2 - OPI Panel Cutout

2.00 [50.8]

4.29 [109.0]

PANEL ASSEMBLY

7.91 [200.8]

J10

1
2
3
4

J1
NEUT
115V
GND

GND (E4)
Vreg (E3)
Tx (E2)
Rx (E1)

IIS
OPERATOR INTERFACE

INDUSTRIAL INDEXING SYSTEMS
Victor, New York
Serial Number
Part Number

MOUNTING BRACKET
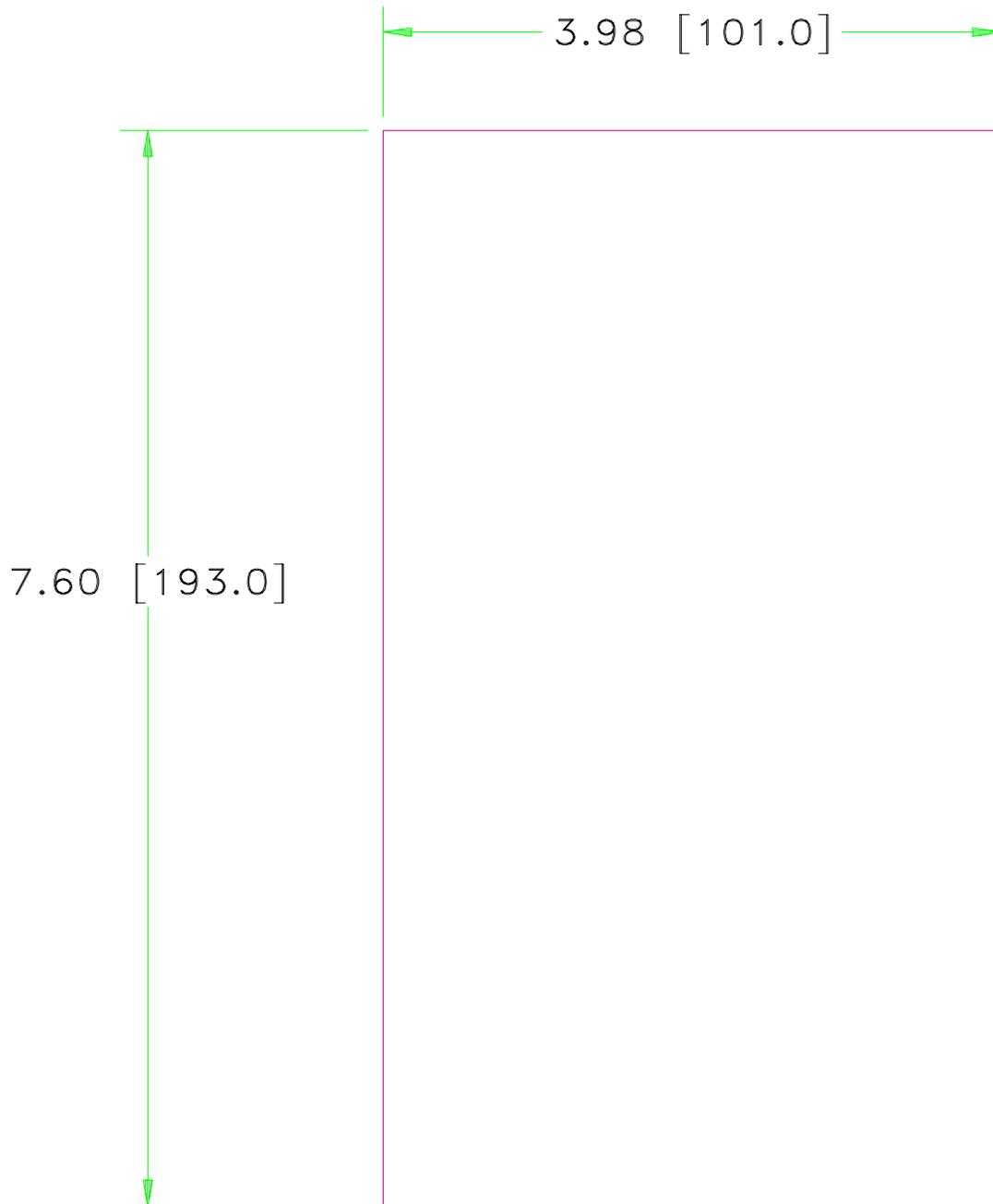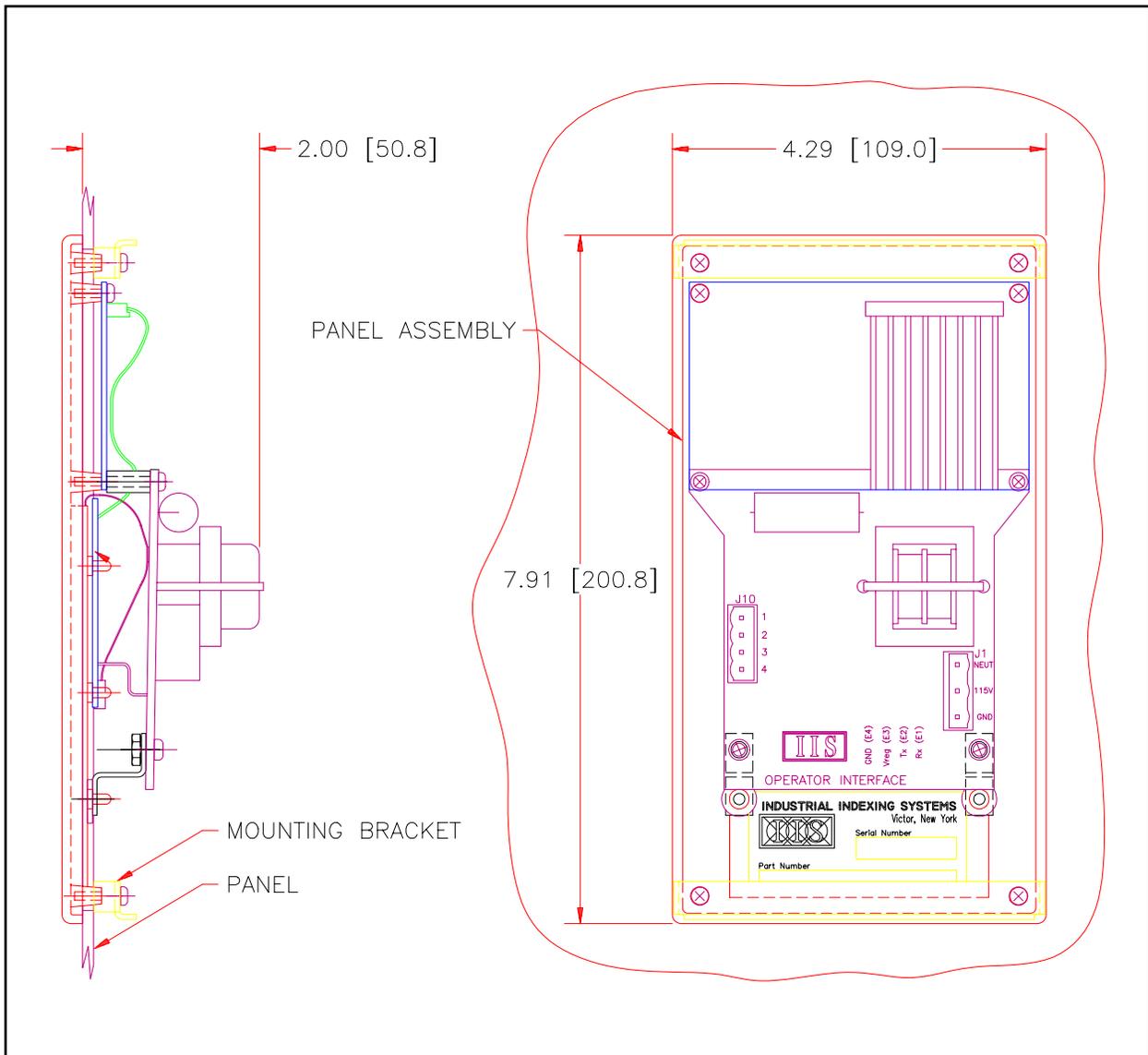
PANEL

Figure 1.3 - OPI Assembly

1.2   Power-on Setup

The Power-on Setup procedure is used to cofigure the baud rate, data format and display contrast of the OPI-7X. Three keys are used to do this configuration '1', '2' and '3'.

To perform the Power-on Setup follow these steps:

1. Disconnect the power from the OPI-7X.

2. Hold down any key and apply power to the OPI-7X.

3. The version of software in the OPI-7X will be displayed for a few seconds after which the contrast can be adjusted.

4. Set the desired contrast using the '1' and '2' keys. Press '3' when the display is at the desired contrast.

5. Set the baud rate using the '1' and '2' keys. Press '3' when the desired baud rate is displayed.

6. Set the desired format using the '1' and '2' keys. When the desired format is displayed press '3'.

1.3   Default Configuration

The default configuration of the OPI-X is as follows:

- operates at 9600 baud, 8 data bits, 1 stop bit, no parity
- contrast is set to optimal for a 90° viewing angle
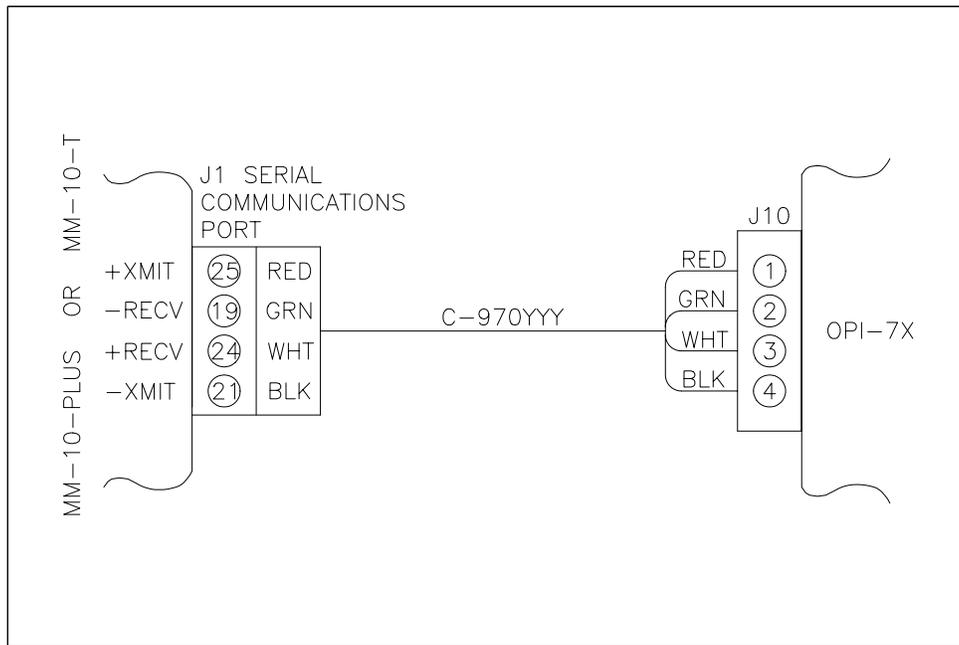
1.4   Interconnect

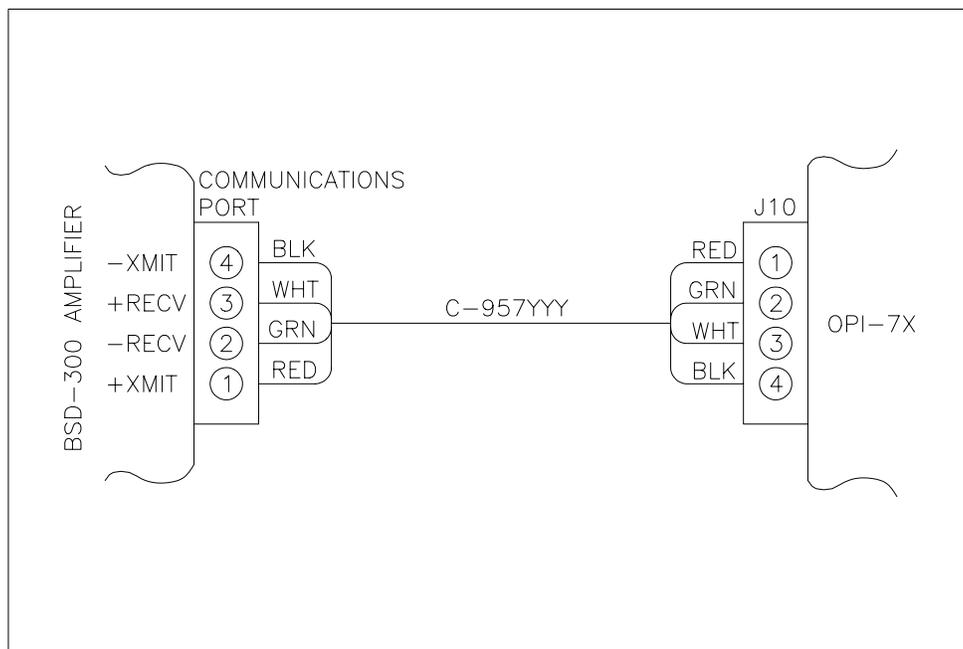FIGURE 1.4 INTERCONNECTION OPI-7X TO MM-10-PLUS OR MM-10-T

FIGURE 1.5 INTERCONNECTION OPI-7X TO BSD-300 AMPLIFIER
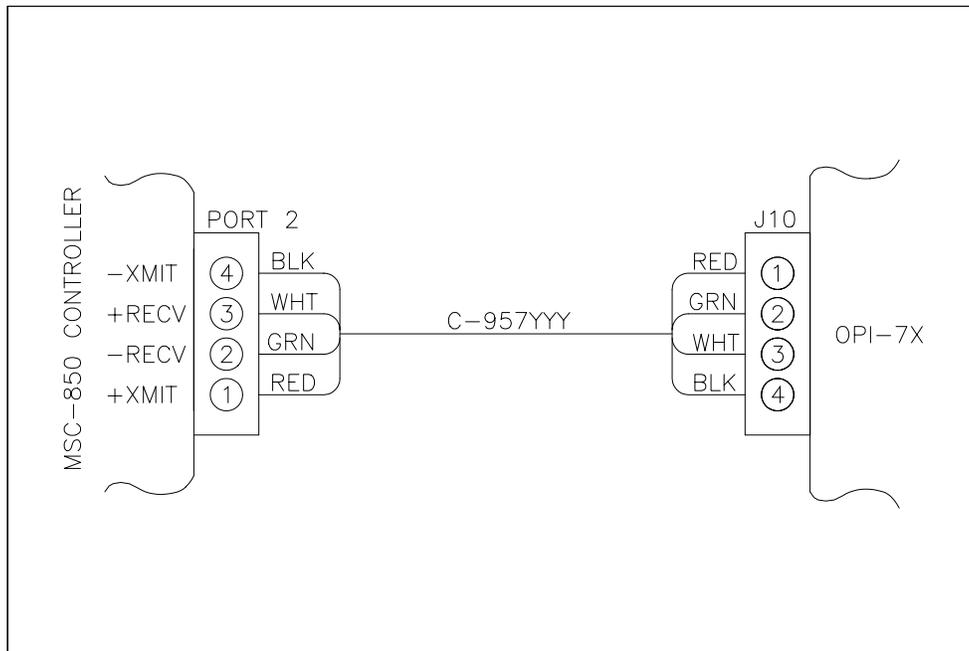
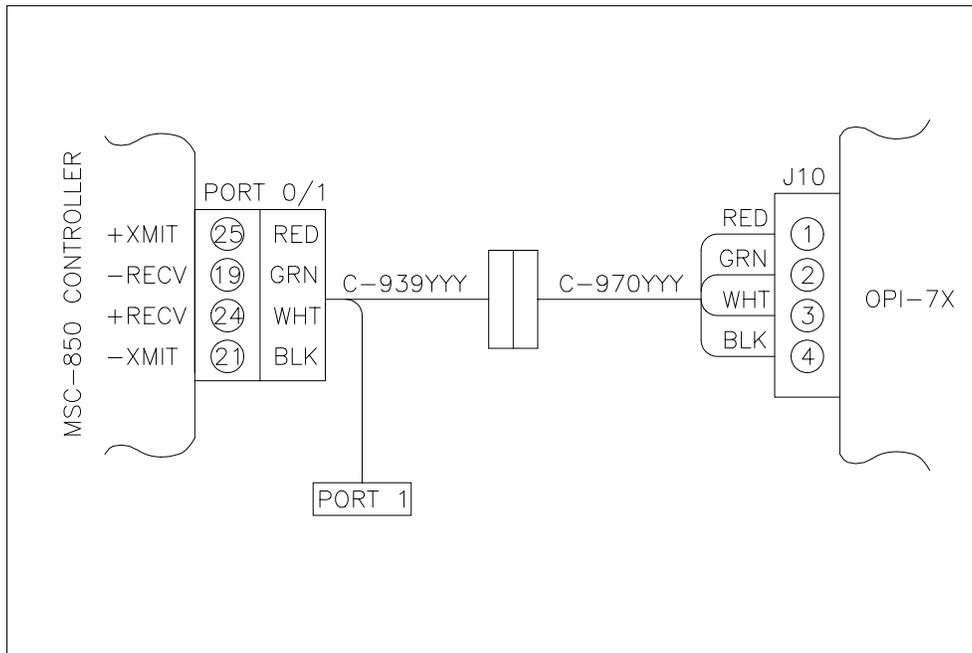FIGURE 1.6 INTERCONNECTION OPI—7X TO MSC—850

FIGURE 1.7 INTERCONNECTION OPI—7X TO MSC—850
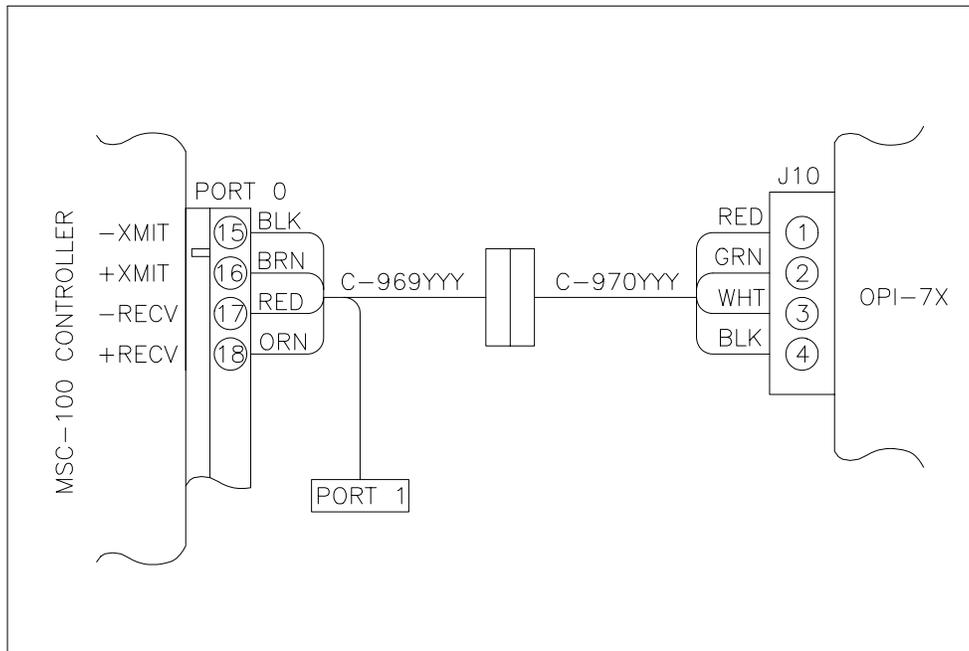
FIGURE 1.8 INTERCONNECTION OPI-7X TO MSC-100
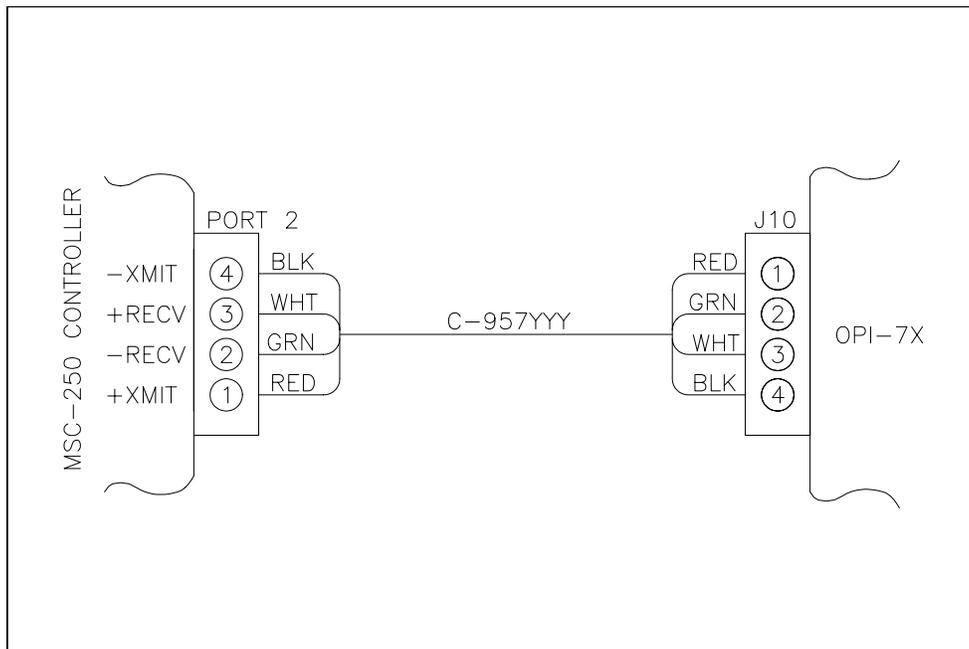
FIGURE 1.9 INTERCONNECTION OPI-7X TO MSC-250
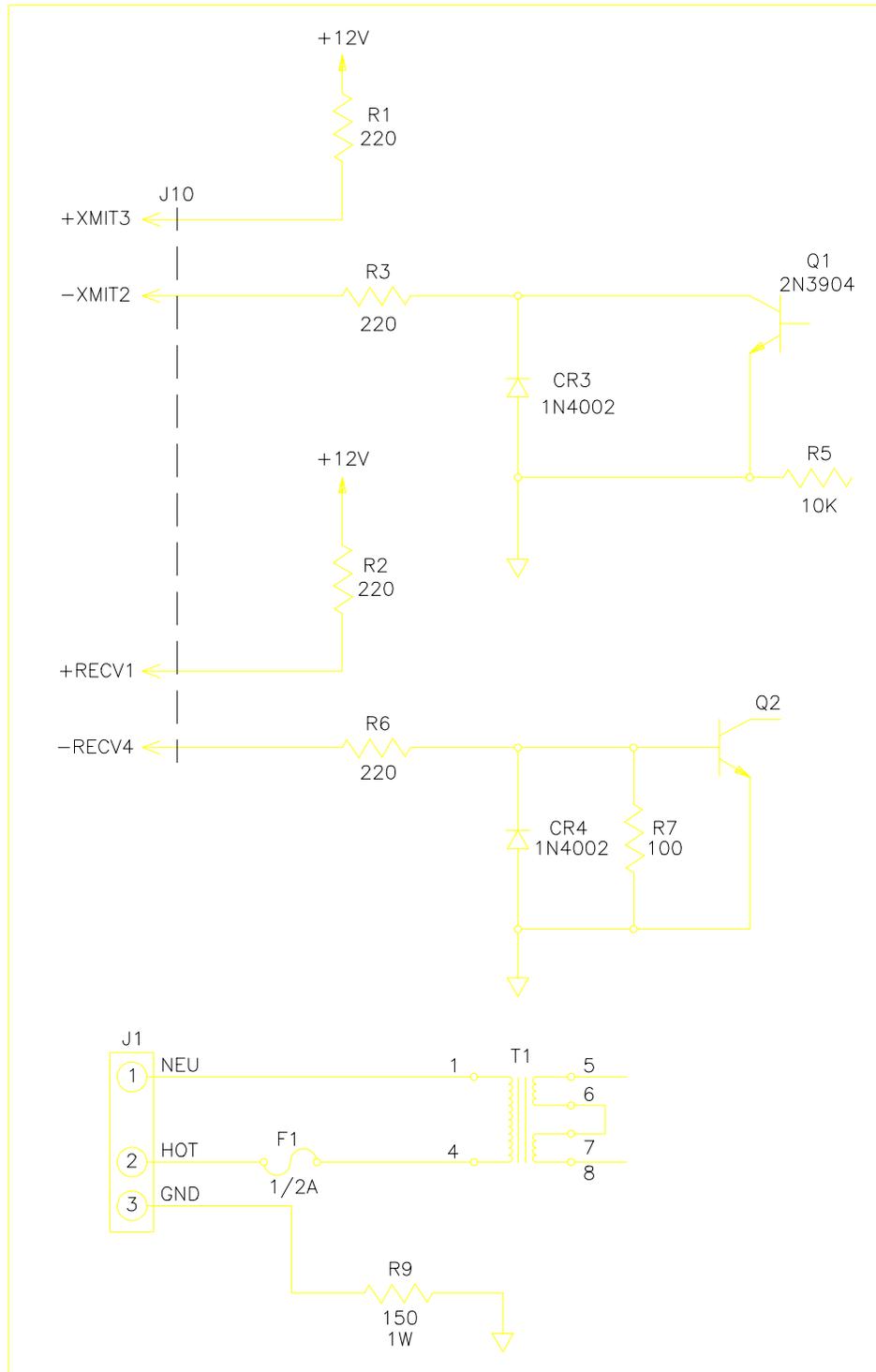
FIGURE 1.10   CTL-70 CONNECTION SCHEMATIC

Section 2 - Programming the OPI-7X

2.1   Introduction

The OPI-7X is an easy to use and easy to program interface to the MSC family of controllers. The programming examples, which follow, are written using the IIS MACROPROGRAM language and demonstrate how to prepare the OPI-7X to accept commands, how to control the display, methods of displaying prompts, checking and verifying user input, and error handling techniques. Prior to writing Macroprograms for the OPI, the user should become familiar with the Macroprogram instructions detailed in **Table 2.1**.

This section of the manual discusses topics including MSC to OPI communications and controlling the display. The final section in this chapter contains a complete source code listing for a macroprogram, which will use all of the techniques discussed.

Important notes about programming the OPI:

1.   **Table 3.1** in Appendix C is a command code summary chart. It lists the various escape sequences which when transmitted to the OPI in the form of an ASCII string become a command to the OPI. Note the length of time required to execute each command. It is possible to overrun the 32-character buffer if sufficient time is not allowed between commands. When reviewing the sample Macroprogram provided, note the use of programmed delays to prevent receiver overrun in the OPI.

2.   The OPI-7X has a backlight, which may enhance the readability of the display. You may want to experiment in the environment in which it will be used to determine whether or not it will be needed.

3.   The 'X' in OPI-7X is a number representing either the standard IIS bezel or an alternate. A value of 70 represents the standard IIS bezel; any other value represents a deviation from the standard IIS bezel.

The following is a list of Macroprogram instructions used in programming the OPI:

INSTRUCTION	DESCRIPTION	PARAMETERS

port_set	open & initialize the selected communication port	port # (0 or 2)
		baud rate: 1200
		                 2400
		                 4800
		                 9600
		protocol:
		0: 1 stop bit, no parity, xon/xoff disabled
		1: 2 stop bits, no parity, xon/xoff disabled
		2: 1 stop bit, even parity, xon/xoff disabled
		3: 2 stop bits, even parity, xon/xoff disabled
		4: 1 stop bit, odd parity, xon/xoff disabled
		5: 2 stop bits, odd parity, xon/xoff disabled
		6: reserved
		7: reserved
		8: 1 stop bit, no parity, xon/xoff enabled
		9: 2 stop bits, no parity, xon/xoff enabled
		10: 1 stop bit, even parity, xon/xoff enabled
		11: 2 stop bits, even parity, xon/xoff enabled
		12: 1 stop bit, odd parity, xon/xoff enabled
		13: 2 stop bits, odd parity, xon/xoff enabled

print	print the ASCII string to the port declared in the last 'port_set' instruction	text label
print_num	print the output value to the port declared in the last 'port_set' instruction	length: no. characters decimals: no. of decimal places value: variable or constant to be displayed
input	read numeric data from the port declared in the last 'port_set' instruction	text string: message to be displayed length: maximum number of characters to be input decimals: no. of decimal places in the value value: address of entered value user flag: flag indicating input is complete
stop_input	terminate any active 'input' instruction and clear input buffer of characters	
if_char	branch to specified address if input is sensed at specified port	port #: 0 or 2 address label: branch address
if_no_char	branch to specified address if input is not sensed at specified port	port #: 0 or 2 address label: branch address
text	defines an ASCII string of characters for use with the 'print' and 'input' instructions	string: an array of characters enclosed in quotes

Table 2.1 - Macroprogram Instructions

2.2   Initiating Communications Between the MSC and the OPI-7X

Before any messages can be displayed on the OPI-7X screen, and before the MSC can receive any input from the OPI-7X, the programmer must open and initialize a communications port. The following example specifies a baud rate of 9600, port 2(current loop port on the MSC-850), 1 stop bit, no parity and xon/xoff. See the **'port_set'** instruction in Table 2.1.

> EXAMPLE:
>
> port_set   2,9600,8   port 2, baud 9600,protocol 8

2.3   Controlling the OPI-7X Display

2.3.1   Special characters for controlling the display

Certain ASCII values represent special characters, which can be embedded in character strings to effectively manage the display. (A character string is a series of characters, which make up messages, or prompts, to be displayed on the screen.) Consult the MacroPro^TM manual or the Macroprogram Davelopment System manual IB-11c001 for a complete description of the Macroprogram Instruction set. ASCII character values are always enclosed in <> to distinguish them from other characters.

EXAMPLE:

'<13><10>'   Carriage Return & Line Feed - These characters, when encountered in a character string, will position the cursor at the first position on the following line in the display area.

'<27>'   Escape - Informs the OPI-7X that the characters immediately following it are terminal control codes.

The OPI-7X has a four line 16 characters per line display. The special function keys (F1-F8) may be used to create a menu system. A sample menu might look something like this:

> F1 CHG SPEED
> F2 CHG AC/DC
> F3 CHG DIRECTION
> F4 CHG DISTANCE

Note: No line is longer than 16 characters.

The ASCII strings to create this menu and the instructions to display it might look like this:

```
EXAMPLE:

menu1        text        "<27>EF1  CHG  SPEED<13><10>F2  CHG  AC/DC<13><10>"
menu2        text        "F3  CHG  DIRECTION<13><10>F4  CHG  DISTANCE"

             print       menu1
             print       menu2
```

The first instruction **'print   menu1'** would act as follows:

1.   The characters '<27>E' embedded in the string 'menu1' clear the screen and move the cursor to the upper left position on the screen.

2.   The string **F1 CHG SPEED** is displayed at the current cursor position.

3.   The characters '<13><10>' are the ASCII Carriage Return and Line Feed values which, when encountered in a text string, cause the cursor to be positioned at the first character on the next line.

4.   **F2 CHG AC/DC** would be displayed on the second line.

5.   The cursor is moved to the beginning of the third line by the characters '<13><10>' (Carriage Return and Line Feed).

The second instruction **'print   menu2'** would act as follows:

1.   **F3 CHG DIRECTION** is displayed on line three.

2.   Another Carriage Return and Line Feed is encountered causing the cursor to be positioned on the first character in the fourth line.

3.   **F4 CHG DISTANCE** is displayed on line four.

2.3.2   Defining the Keypad and Returned Key Values

The OPI-7X will ignore the commands used to assign an ASCII character to each key on the keypad and the one for defining the keypad character set. The keypad layout and the corresponding returned decimal values follow:

```
        ASCII Characters        Decimal value returned
        which define keys         when key is pressed

         F1   F2 F3   F4              65 66 67 68
         F5   F6 F7   F8              69 70 71 72
         BS    1  2    3              8  49 50 51
        NEXT   4  5    6              78 52 53 54
        PREV   7  8    9              80 55 56 57
          -    .  0  ENTER            45 46 48 13
```

2.3.3   Positioning the cursor

The cursor may be turned **ON** and **OFF** and may be either block or underline. A good rule to follow is to turn the cursor OFF when waiting for single keystroke input and ON when waiting for multi-character input. The following example demonstrates how to turn the cursor ON and OFF.

```
EXAMPLE:

cursor_on          text        "<27>wC"        turn cursor on
cursor_offl        text        "<27>wA"        turn cursor off

                   print       cursor_on
```

The following steps show how to position the cursor to a selected X/Y position on the display:

  curs_pos   text   "<27>l "   move cursor to x,y

Note: It is important to leave 1 blank space after the letter 'l' in the string curs_pos. A numeric value will be placed in that space to define the new cursor position.

The display X/Y coordinates are as follows:

```
Y   _____
    0

    1

    2

    3
X   _____
    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```
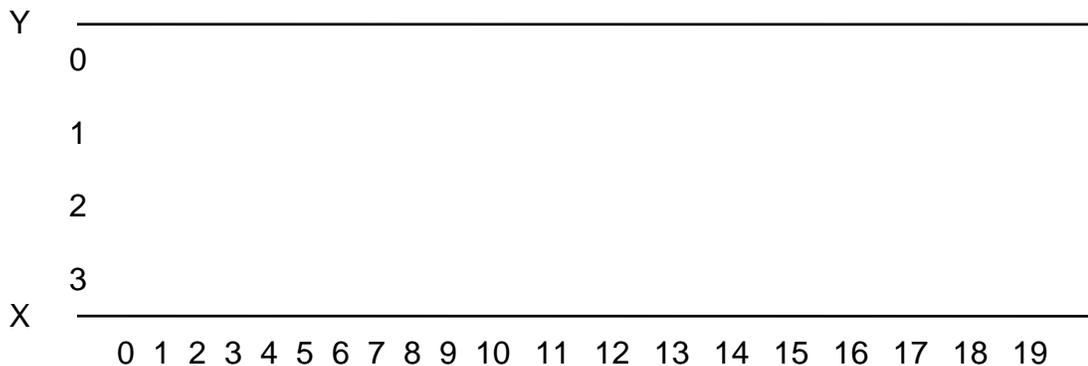
Step 1.   Determine the x and y position for cursor placement. (X is a value between 0 and 15, and Y is a value between 0 and 3)

Step 2.   Multiply the Y value by 16:

               let   y=y*16

Step 3.   Add x and y:

               let   x_y_sum=x+y

Step 4.   Add 64 to the value calculated in step 3:

        let   **x_y_sum=x_y_sum+64**

Step 5.   Move this value into the third character of the string curs_pos defined above:

        let_byte   **curs_pos**   **[2]=x_y_sum**

Step 6.   Execute the instruction to move the cursor to the selected position within the display:

        print   curs_pos

2.3.4   Backlight

The OPI-7X has an optional backlight, which, depending on the environment in which it is used may make it easier to read the display. The following example demonstrates how to turn this light ON and OFF.

```
EXAMPLE:

light_on        text    "<27>V"      turn backlight on
light_off       text    "<27>V"      turn backlight off

                print   light_on
                print   light_off
```

2.3.5   Prompting, Data Entry and Error Handling

In prompting the operator to enter a numeric value, the following suggestions may be followed:

1. Display a prompt that clearly instructs the operator on what is to be entered. If possible, display the value currently in memory and the range of acceptable values.
2. Check the data entered against the minimum and maximum values allowed. If the data value entered is out of range, display an error message and the correct range of values.

3. Leave the error message on the display until the operator presses a function key indicating he has seen the message and is ready to enter a new value.

   The sample Macroprogram in **Section 2.4** demonstrates prompting, data entry and error handling routines, which follow the above rules.

2.3.6   Other considerations

When the OPI-7X is first powered on, a certain delay is required before it can respond to input data or key process. This delay is approximately 300 milliseconds. The Command Code Summary chart in the appendix shows the approximate execution time required by the commands.

2.4   Sample Macroprogram


```
!
!
!   PROGRAM:      An example using an OPI-7X device & the MSC-850
!
!   DESCRIPTION:  This program will prompt the operator to enter speed, accel/
!                 decel rate and a absolute position (as motor turns xx.xx)
!                 and then execute a move to that position with the parameters
!                 specified.
!
!                 As each entry is made, the program will verify that it is within the
!                 proper range and will alert the operator if not.
!

              msc_type      850
              declare       ON
```

AXIS_1	equ	1	
DOWN_1	equ	93	
BUSY_1	equ	94	
TIMER_1	equ	72	
BITS_TURN	equ	4096	
DEF_SPEED	equ	10	
MIN_SPEED	equ	1	
MAX_SPEED	equ	2000	
DEF_ACDC	equ	100	
MIN_ACDC	equ	5	
MAX_ACDC	equ	800	
DEF_TURNS	equ	0	
MIN_TURNS	equ	-9999	! -99.99 turns
MAX_TURNS	equ	9999	! +99.99 turns
ENTER	equ	255	!Enter Character or Carriage Return
NEXT	equ	78	!Next Character
PREV	equ	80	!Previous Character
F1	equ	65	!F1 Function Key
F2	equ	66	!F2 Function Key
F3	equ	67	!F3 Function Key
F4	equ	68	!F4 Function Key
F5	equ	69	!F5 Function Key
F6	equ	70	!F6 Function Key
F7	equ	71	!F7 Function Key
F8	equ	72	!F8 Function Key
MENU_1_1	text	"F1 SETUP<13><10>"	
MENU_1_2	text	"F2 RUN<13><10>"	
MENU_1_3	text	"<13><10>"	
MENU_1_4	text	"F4 STOP PROG<13><10>"	
SPEED_IN	text	"SPEED: "	
ACDC_IN	text	"AC/DC: "	
TURNS_IN	text	"TURN: "	
POSITIONING	text	"POSITIONING ..."	
ERR_1	text	"ENTRY:<13><10>"	
ERR_2	text	" OUT OF RANGE<13><10>"	
ERR_3	text	" TO<13><10>"	

ERR_4   text   " PRESS ANY KEY "

!***
!

CRLF	text	"<13><10>"	!Carriage Return and Line Feed
CUR_POS	text	"<27>I "	!Cursor Position
CLR_SCREEN	text	"<27>E"	!Clears The Screen
CUR_HOME	text	"<27>H"	!Sends Cursor to Home Position
CUR_OFF	text	"<27>wC"	!Turns the Cursor Off
CUR_ON	text	"<27>wA"	!Turns the Cursor On
BACKLITE	text	"<27>V"	!Backlite on
NULL	text	""	!Null string Character
CUR_RIGHT	text	"<27>C"	!Move cursor right one space
CUR_DOWN	text	"<27>B"	!Move cursor down vertically !one space

!Program Variable Declarations

speed	integer
acdc	integer
turns	integer
key	integer
temp	integer
row	integer
col	integer
turn_bits	integer
whole_turn	integer
frac_turn	integer
r_value	integer
result	integer
new_result	integer
diff	integer
time	integer
time_1	integer
err	integer
ctr	integer

!***
! -------- PROGRAM SETUP ---------
!***

	drive_on	AXIS_1
	set_gl_ccw	AXIS_1
	gosub	Opi_setup
	let	speed=DEF_SPEED
	let	acdc=DEF_ACDC
	let	turns=DEF_TURNS

```
!*********************************************************************************
! -------- MAIN MENU ---------
!*********************************************************************************
main            print           CLR_SCREEN
                print           MENU_1_1
                print           MENU_1_2
                print           MENU_1_3
                print           MENU_1_4

main_key        gosub           ret_f_key
                if              key=F1,speed
                if              key=F2,run
                if              key=F4,exit
                goto            main_key


!*********************************************************************************
! ---------- SETUP SPEED ----------
!*********************************************************************************
speed           print           CLR_SCREEN
                print           SPEED_IN
                print_num       4,0,speed
                print           CRLF
                let             temp=speed

                input           NULL,7,0,temp,ENTER
speed_wt        if_flag_off     ENTER,speed_wt
                if              temp>MAX_SPEED,speed_err
                if              temp<MIN_SPEED,speed_err
                let             speed=temp
                goto            set_acdc

speed_err       gosub           err_screen
                let             row=0
                let             col=9
                gosub           cursor
                print_num       7,0,temp

                let             row=2
                let             col=0
                gosub           cursor
                print_num       4,0,MIN_SPEED

                let             row=2
                let             col=12
```

```
                gosub           cursor
                print_num       4,0,MAX_SPEED
                gosub           press_key
                goto            speed




!************************************************************************************************
! ---------- SETUP ACCEL/DECEL RATE ----------
!************************************************************************************************
set_acdc        print           CLR_SCREEN
                print           ACDC_IN
                print_num       4,0,acdc
                print           CRLF
                let             temp=acdc

                input           NULL,7,0,temp,ENTER
acdc_wt         if_flag_off     ENTER,acdc_wt
                if              temp>MAX_ACDC,acdc_err
                if              temp<MIN_ACDC,acdc_err
                let             acdc=temp
                goto            set_turns

acdc_err        gosub           err_screen
                let             row=0
                let             col=9
                gosub           cursor
                print_num       7,0,temp

                let             row=2
                let             col=0
                gosub           cursor
                print_num       3,0,MIN_ACDC
                let             row=2
                let             col=13
                gosub           cursor
                print_num       3,0,MAX_ACDC
                gosub           press_key
                goto            set_acdc




!************************************************************************************************
!
! ---------- SETUP POSITION IN TURNS ----------
!************************************************************************************************
set_turns       print           CLR_SCREEN
                print           TURNS_IN
                print_num       8,2,turns
                print           CRLF
```

```
                    let                 temp=turns

                    input               NULL,8,2,temp,ENTER
turns_wt            if_flag_off         ENTER,turns_wt
                    if                  temp>MAX_TURNS,turns_err
                    if                  temp<MIN_TURNS,turns_err
                    let                 turns=temp
                    goto                main


turns_err           gosub               err_screen
                    let                 row=0
                    let                 col=9
                    gosub               cursor
                    print_num           7,2,temp

                    let                 row=2
                    let                 col=0
                    gosub               cursor
                    print_num           6,2,MIN_TURNS

                    let                 row=2
                    let                 col=10
                    gosub               cursor
                    print_num           6,2,MAX_TURNS
                    gosub               press_key
                    goto                set_turns



!**********************************************************************************************
! -------- RUN MODE ---------
!**********************************************************************************************
run                 print               CLR_SCREEN
                    print               POSITIONING
                    set_speed           AXIS_1,speed
                    set_ac_dc           AXIS_1,acdc
                    gosub               turns_to_bits

                    position            AXIS_1,turn_bits
run_busy            if_stat_on          DOWN_1,fault
                    if_stat_on          BUSY_1,run_busy
                    goto                main



!**********************************************************************************************
! -------- CONVERT TURNS TO BITS --------
!**********************************************************************************************
turns_to_bits       let                 turn_bits=0
```

```
                 if        turns=0,ttb_end
                 let       whole_turn=turns/100
                 let       turn_bits=whole_turn*BITS_TURN
                 let       whole_turn=whole_turn*100
                 let       frac_turn=turns-whole_turn
                 let       r_value=frac_turn*BITS_TURN
                 gosub     round              ! intentionally 'round' twice
                 gosub     round              ! due to turns to hundredths
                 let       turn_bits=turn_bits+r_value
ttb_end          return_sub
```

```
!*******************************************************************************
! -------- ROUND function --------
!*******************************************************************************
round            let       result=r_value/10
                 let       new_result=result*10
                 let       diff=r_value-new_result
                 let       r_value=result
                 if        diff<5,end_round
                 let       r_value=result+1
end_round        return_sub
```

```
!*******************************************************************************
! -------- EXIT PROGRAM ---------
!*******************************************************************************
exit             f_decel   AXIS_1
                 let       time=200
                 gosub     pause
                 drive_off AXIS_1
                 print     CLR_SCREEN
                 sys_return
```

```
!*******************************************************************************
! -------- AXIS FAULT ENCOUNTERED ---------
!*******************************************************************************
fault            f_decel   AXIS_1
                 let       time=200
                 gosub     pause
                 drive_on  AXIS_1
                 goto      main
```

```
!**********************************************************************************
! -------- RETURNS THE SELECTED FUNCTION KEY ---------
!**********************************************************************************
ret_f_key          print           CUR_OFF

                   input           NULL,0,0,key,ENTER
rfk_wait           if_flag_off     ENTER,rfk_wait

                   let             err=0
                   if              key=F1,rfk_ok
                   if              key=F2,rfk_ok
                   if              key=F3,rfk_ok
                   if              key=F4,rfk_ok
                   if              key=NEXT,rfk_ok
                   if              key=PREV,rfk_ok
                   if              key=ENTER,rfk_ok
                   goto            ret_f_key

rfk_ok             print           CUR_ON
                   return_sub



!**********************************************************************************
! -------- PRESS ANY KEY TO CONTINUE --------
!**********************************************************************************
press_key          stop_input
                   input           NULL,0,0,key,ENTER
pk_wait            if_flag_off     ENTER,pk_wait
                   return_sub



!**********************************************************************************
! -------- POSITION CURSOR FUNCTION ---------
!**********************************************************************************
cursor             print           CUR_HOME

                   let             ctr=0
c_next_col         if              ctr=col,c_row
                   print           CUR_RIGHT
                   let             ctr=ctr+1
                   goto            c_next_col

c_row              let             ctr=0
c_next_row         if              ctr=row,c_done
                   print           CUR_DOWN
                   let             ctr=ctr+1
                   goto            c_next_row
```

```
c_done              return_sub



!***************************************************************************************
! -------- TEMPLATE FOR ERROR DISPLAY ---------
!***************************************************************************************
err_screen          print           CLR_SCREEN
                    print           ERR_1
                    print           ERR_2
                    print           ERR_3
                    print           ERR_4
                    return_sub



!***************************************************************************************
! -------- TIMERS ---------
!***************************************************************************************
pause               set_tmr         TIMER_1,time_1
pausing             if_tmr_on       TIMER_1,pausing
                    return_sub



!***************************************************************************************
!
!   NAME:           Opi_setup
!
!   DESC:           Used to initialize the OPI to the proper settings and display
!                   initial message.
!
!***************************************************************************************

Opi_setup           no_op
                    port_set        2,9600,8
                    print           CLR_SCREEN
                    print           CUR_HOME
                    print           CUR_OFF
                    print           BACKLITE
                    let             time_1=200
                    gosub           pause
                    return_sub
!***************************************************************************************
```

Section 3 - Appendices

APPENDIX A. ASCII Conversion Table

The following tables provide information for decimal-hexadecimal ASCII conversions.

DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	00	NUL	32	20	SP	64	40	@	96	60	'
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	I
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	'	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	]	123	7B	{
28	1C	FS	60	3C	<	92	5C		124	7C	
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	VS	63	3F	?	95	5F		127	7F	DEL

APPENDIX B. Terminal Code Commands

The OPI-7X terminal is controlled through the use of the **'print'**, **'print_num'**, and **'input'** instructions. Control of the display is accomplished by creating text strings and using the **'print'** instruction to transmit them to the OPI-7X. These text strings may contain ASCII control codes, which are used to control the display.

Backspace - 08h

Causes a non-destructive backspace, i.e., characters are not erased as the cursor is backspaced over them. With auto wrap mode off, the backspace stops at the left edge of the current display line. With auto wrap mode on, the cursor will wrap to the last position on the previous line. The command is ignored if the cursor is at the home position. See also the *Delete* command.

Horizontal Tab - 09h

Moves the cursor right to the next tab column. The tap spacing is every four columns.

With auto wrap on, the cursor will wrap down to the first column in the line below when it is tabbed beyond the last column in the current line. If auto wrap is off, the cursor will stop at the end of the current line. If auto scroll and auto wrap are both on, then the display will scroll up as the cursor is tabbed beyond the last column in the last line.

Line Feed - 0Ah

Moves the cursor down one line without changing its horizontal position. When auto scroll mode is on and a line feed is performed on the last line, the display will scroll up with the horizontal cursor position unaltered.

Vertical Tab - 0Bh

Performs the same function as *Line Feed*.

Form Feed - 0Ch

Performs the same function as *Line Feed*.

Carriage Return - 0Dh

Moves the cursor to left edge of the display on the current line. If auto line feed mode is on, then the cursor moves to the left edge of the next line. If auto scroll and auto line feed are both on, a carriage return on the last line will cause the display to scroll up and the cursor to be positioned at the left edge of the last line.

XON - 11h

Enables the OPI-X to transmit keys pushed after receiving an XOFF. XON is used to re-enable OPI-X transmission after an XOFF has disabled it, allowing handshaking with the host system.

XOFF - 13h

Disables all OPI-X transmission except for information requested via *the Query Status* command (ESC W). After receiving an XOFF command, the OPI-X stores characters typed on the keypad in a transmit buffer. These characters will be transmitted when an XON is received. If the buffer becomes full before an XON is received, additional characters, which are typed, will be ignored.

Delete - 7Fh

Delete works in the same way as *Backspace* (08H), except that characters are erased as the cursor moves over them.

Cursor Up - ESC A

Moves the cursor up one line without changing its horizontal position. Has no effect if the cursor is on the first line.

Cursor Down - ESC B

Moves the cursor down one line without changing its horizontal position. Has no effect if the cursor is on the last line.

Cursor Right - ESC C

Moves the cursor right one space without changing its vertical position. Has no effect if the cursor is at the right-most position on the current line.

Cursor Left - ESC D

Moves the cursor left one space without changing its vertical position. Has no effect if the cursor is at the left-most position on the current line.

Clear Screen - ESC E

Clears the display and moves the display cursor to home (the left-most position in the top line of the display).

Cursor Home - ESC H

Moves the cursor to the home (top left) position on the display.

Set Cursor Position - ESC I#

Positions the cursor to the specified location. The # is in the range from 64 to 127. For example, the string:

ESC I104

sets the cursor to row 2 (third row) and column 8 (ninth column). See **Table 2.1** for a complete list of valid codes and cursor positions. (Note that rows are numbered 0 to 3, starting at the top, and columns are numbered 0 to 15, starting at the left.)

Erase to End of Screen - ESC J

Erases from the current cursor position to the end of the screen. The cursor position is unchanged.

Erase to End of Line - ESC K

Erases all displayed characters from the current cursor position to the end of the line. The cursor position is unchanged.

Set Contrast - ESC L

This command sets the display contrast. It has the form *ESC L #,* where # is in the range of 40h to 7Fh ('@' to DEL). The smaller the ASCII value of the character, the lower the contrast. The higher the ASCII value of the character, the higher the contrast.

Reset Terminal - ESC M

Resets the OPI-X to its power-up state. This includes clearing all input and output buffers and the display, and resetting all parameters to the default configuration.

Query Version - ESC N

This tells the OPI-X to transmit its software version to the host. The version will consist of four ASCII characters in the format *vx.y*, where x and y are single ASCII digits.

Auto Wrap Mode - ESC

The auto wrap determines what happens when the cursor moves past the end of a line. With auto wrap off, the cursor stays at the last position in the line. With auto wrap on, the cursor moves down to the first position in the next line.

If the cursor moves past the end of the last line, and auto wrap is on, then the action depends on the auto scroll mode. If auto scroll is off, the cursor will wrap to the first position of the line, but the display will not scroll. Otherwise, the display will scroll, and the cursor will return to the first position in the last line.

Valid values for # are:

      r - auto wrap off
      R - auto wrap on

Auto Scroll Mode - ESC

Auto scroll mode determines what happens when the cursor moves past the end of the last line. With auto scroll off, the cursor will stay in the last position. With auto scroll on, the display scrolls (i.e. every lines moves up, and the last line becomes blank), and the cursor moves to the first position in the last line.

Valid values for # are:

      s - auto scroll off
      S - auto scroll on

Auto Line Feed Mode - ESC

With auto line feed off, when a carriage return is received the cursor returns to the first position in the current line. With auto line feed on, the cursor moves to the first position in the next line, i.e. it acts as if both a carriage return and a line feed had been received.

Valid values for # are:

      t - auto line feed off
      T - auto line feed on

Backlight On/Off - ESC

This command turns the backlight on and off. Valid values for # are:

      v - backlight off
      V - backlight on

Query/Set Status - ESC W/w

The Query Status command (ESC W) returns the timeout status and the number of characters in the keyboard buffer of the OPI-X. The timeout status is contained in bit 5 of the returned status byte: bit 5 is set if the display has timed out, and cleared otherwise. Bits 0 - 4 contain the number of characters in the keyboard buffer waiting to be transmitted (0 to 16). Bit 7 is always 0, and bit 6 is always 1. Because of bits 6 and 7, the status byte returned will always be an ASCII character, ranging from "@" to "Delete".

The Set Status command controls three items: underline cursor, blinking-block cursor, and key repeat. The command form is "ESC w #" where "#" is a single letter A through H. The table below shows what each letter selects. Note that it is possible to have the underline and blinking-block cursor at the same time, or to have no cursor at all.

The default state upon power-up is equivalent to "ESC w F"; underline cursor, key repeat, no blinking-block cursor.

SET STATUS COMMAND OPTIONS

Command Letter	Underline Cursor	Blinking Block Cursor	Key Repeat
A	no	no	no
B	no	no	yes
C	no	yes	no
D	no	yes	yes
E	yes	no	no
F	yes	no	yes
G	yes	yes	no
H	yes	yes	yes

The Query Status command (ESC W) returns a character indicating the number of characters presently in the OPI-X transmit buffer (0 to 20). **Table 2.1** shows what character is returned for each number of characters in the transmit buffer.

Query Cursor Position - ESC X

Returns the cursor position. The value returned is in same format discussed in the *Set Cursor Position* (ESC I) command.

Query Character - ESC Y

Returns the character at the current cursor position.

Set Shift Mode - ESC c #

The shift key on the OPI-X can operate in one of two ways:

- FUNCTION MODE, where the shift key stays shifted for one additional key press only

- LOCK MODE, where the shift key stays shifted until it is pressed a second time

To indicate shift status, the cursor will change to a blinking block when shifted, and go back to current cursor status when not shifted. For either mode, this indicator can be enabled or disable, and, if enabled, properly reflects the state of the shift key. Valid values for # are:

    @ - function mode, shift indicator enabled
    A - lock mode, shift indicator enabled
    B - function mode, shift indicator disabled
    C - lock mode, shift indicator disabled

Note that the shift indicator does not affect the shift operation itself, only the status indicator. Also note that any time you are using the blinking block cursor, the shift indicator will not be available.

Save Configuration to EEPROM - ESC i

This command causes all parameter values to be stored to EEPROM. Any existing parameter values in the EEPROM will be overwritten.

Transmit Buffer Flush - ESC k

If the host has transmitted an XOFF to the OPI-X, and the user has pressed any keys, this command will clear the buffer, so that when the host sends XON to the OPI-X, there will be nothing in the buffer for the OPI-X to transmit to the host.

XON/XOFF Mode - ESC l #

This command enables or disables the XON/XOFF operation of the OPI-X. The valid values for # are:

    @ - disable XON/XOFF operation
    A - enable XON/OFF operation

If you disable XON/XOFF operation, then any keys pressed by the user will be sent to the host immediately. If the host sends data fast enough to the OPI-X to fill up the receive buffer, additional characters will be ignored.

User Area Read/Write - ESC m

This command allows you to store your own information (such as serial numbers or parameters) in the OPI-X nonvolatile EEPROM, and then later read them from the terminal. There are two valid values for #:

> @ - read user data
> A   - write user data (followed by data)

The OPI-X can store a maximum of 16 bytes in the user data area.

*READ DATA:*if # = '@', the OPI will transmit the data in the user area to the host in the following format:

> # ....

where # is a character in the range of 40h to 50h, and indicates that 0 to 16 bytes of user data will follow, and '....' is the corresponding number of user bytes. These bytes will be exactly what was originally stored, so they may be any 8-bit value. If # = '@' (0 bytes to follow), then there was no data stored in the user area.

WRITE DATA: to write user data, use the format:

> ESC m A # ....

where # is in the range of 41h to 50h ('A' to 'P'), and indicates that from 1 to 16 bytes of data are to follow, and '....' are the data bytes to be stored. These data bytes may be any 8-bit value.

After the entire string has been received, the OPI will respond by transmitting one character to the host:

06h   -   Acknowledge character (ACK), data stored properly
15h   -   Negative Acknowledge character (NAK), data no stored

The only reason that the data would be stored properly is if there was a hardware failure.

Restore Default Parameters - ESC r

This command will load a set of factory-default values for all parameters (except baud rate and data format) into memory, and write them to EEPROM making them the current power-up settings.

Verify Security Bytes - ESC u # *

This command is used to verify a set of security bytes, which are pre-programmed into the terminal at the factory. # and * can be any two bytes. When ordering a OPI-X from the factory, the customer is given the option to use this feature and will be assigned a two byte security code. When this command is sent, along with two bytes, the terminal will compare them to its internally stored byte pair, and respond with one of the following:

06h   -   Acknowledge character (ACK), Security bytes match
15h   -   Negative Acknowledge character (NAK), Security bytes do not match

Power-On Setup Mode - ESC x#

This command can be used to enable or disable the power-on setup feature. In some cases, it may be desirable to disable the power-on setup in order to protect the current baud rate and data format settings from being changed by the user. Valid values for # are:

  @ - Fully enable power-on setup
  A - Allow contrast adjustment, but do not allow baud rate and data format
        adjustment
  B - Disable entire power-on setup

APPENDIX C. Command Code Summary Chart

The following chart summarizes all of the command codes available to control the OPI-X terminal.

COMMAND	CODE	TIMING (ms)	NOTES & PARAMETERS
display character		0.7	typical
		35	maximum
Backspace	08H	0.8	
Horizontal Tab	09H	0.8	
Line Feed	0AH	0.8	
Vertical Tab	0BH		
Form Feed	0CH		
Carriage Return	0DH	0.8	
XON	11H	0.4	
XOFF	13H	0.4	
Delete	7FH	1.2	
Cursor Up	ESC A	0.8	
Cursor Down	ESC B	0.8	
Cursor Right	ESC C	0.8	
Cursor Left	ESC D	0.8	
Clear Screen	ESC E	4.4	
Cursor Home	ESC H	3.7	
Set Cursor Position	ESC I	1.5	
Erase to End of Screen	ESC J	6.0	if cursor is on row 3
		12.0	if cursor is on row 2
		18.0	if cursor is on row 1
		24.0	if cursor is on row 0

Erase to End of Line	ESC K	3.0 5.8	typical maximum (cursor in column 0)
Reset Terminal	ESC M	300.0	
Query Version	ESC N	0.5	time to load characters into transmit buffer
Auto Wrap Mode	ESC R / r	1.2	r auto wrap off R auto wrap on
Auto Scroll Mode	ESC S / s	1.2	s auto scroll off S auto scroll on
Auto Line Feed Mode	ESC T / t	1.2	t auto line feed off T auto line feed on
Display Backlight Mode	ESC V / v	1.2	v backlight off V backlight on
Query Status	ESC W	0.5	time to load characters into transmit buffer see **Table 2.1** for returned values
Query Cursor Position	ESC X	0.5	time to load characters into transmit buffer see **Table 2.1** for returned values
Query Character	ESC Y	0.5	time to load character into transmit buffer
Set Shift Mode	ESC c #	1.2	# = @ function mode, shift indicator enabled A lock mode, shift indicator enabled B function mode, shift indicator disabled C lock mode, shift indicator disabled
Save Configuration to EEPROM	ESC I	6.0	
Transmit Buffer Flush	ESC k	1.2	
XON/XOFF Mode	ESC l #	1.2	# = @ disable XON/XOFF position A enable XON/XOFF position
User Area Read/Write	ESC m #		timing depends on baud rate
Restore Default Parameters	ESC r	3.0	

Verify Security Bytes	ESC u # *	2.0	# and * are the bytes you want to compare
Power-On Setup Mode	ESC x #	1.2	# = @ Power-on setup fully enabled A Only contrast adjustment allowed B Power-on setup fully disabled

INDUSTRIAL INDEXING SYSTEMS INC.
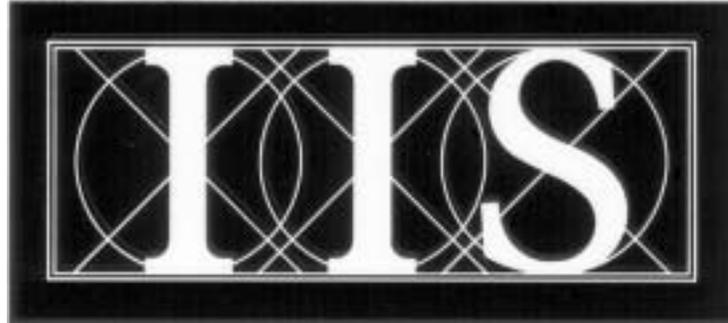
626 FISHERS RUN
VICTOR, NEW YORK 14564

(585) 924-9181
FAX: (585) 924-2169