Table of Contents

1 OptiTracker

1.1 Features

The OptiTracker is a high speed 1, 2, 3, or 4 axis quadrature encoder interface designed for standard IBM-PC/XT/AT or compatibles. The maximum speed is a respectable 1.3 MHz in either quadrature or count modes. In quadrature mode, the inputs, channel A and B, are typical 90° out-of-phase signals and in count mode, the inputs are pulse and direction signals. The counters will track absolute counts to ± 8,388,607. The index pulse for each encoder as well as four additional inputs can be read from the input port.

<u>The features of OptiTracker include:</u>

- 1.3 MHz maximum input frequency in both modes
- 4 - 24 bit presettable counters
- Counter enable/disable input for frequency measurements
- User selectable X1, X2, X3, and X4 quadrature multiplier
- General purpose 8-bit input port

<u>Some Applications of the OptiTracker include:</u>

- Robotics
- Machine Control
- Digital Readout for positioning systems
- Frequency measurement
- Encoder feedback for closed-loop control

1.2 Specifications & Requirements

Electrical Specifications:

Connector	Standard DB-25 female type
Encoder Compatibility	Single ended, open collector
Typical Power Consumption (+5 Vdc)	
1 Axis	0.50 A
2 Axis	0.75 A
3 Axis	1.10 A
4 Axis	1.35 A
Working Temperature range	(32°F ~ 122° F (0° C ~ 50° C)
Warranty	5 years parts and labor

Hardware Specifications

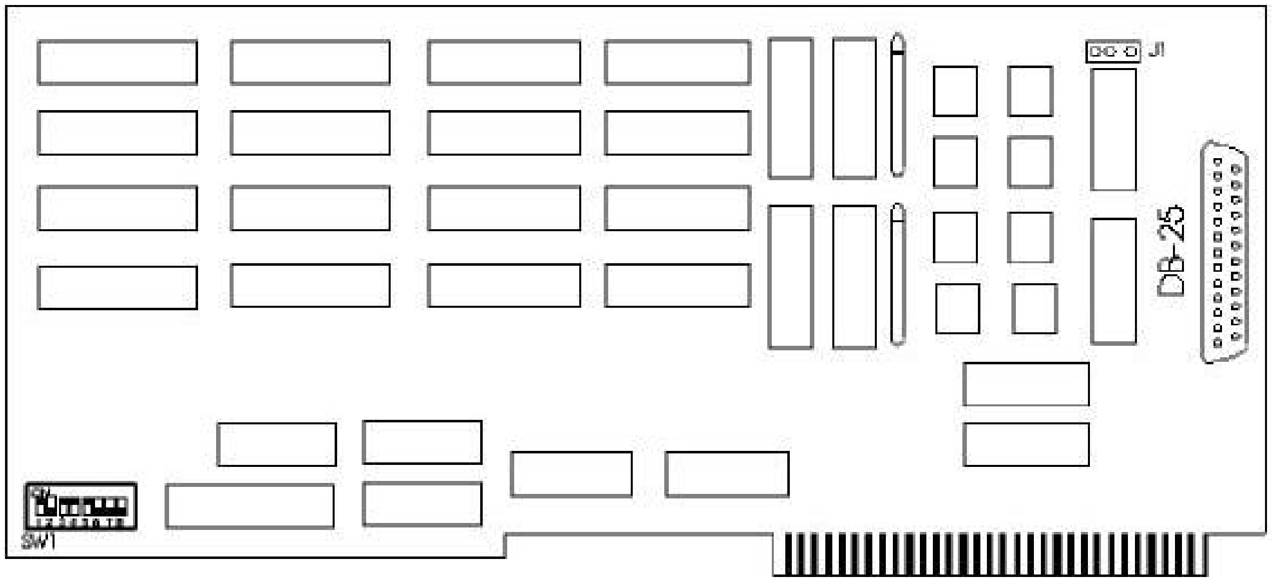


Figure 1 - OptiTracker Mechanical Layout Diagram

Requirements:

- 8 or 16 bit IBM PC / AT expansion slot

2 - User Information

2.1 Installation

1) Turn off the computer.

2) Set the dip switches (SW1-5) on the card so that the selected I/O address does not conflict with other cards in the system. (The default, 640, setting is usually acceptable). Refer to Appendices A & B for more information.

3) Set the input mode dip switch in either quadrature mode (SW8-OFF) or count mode (SW8-ON). If the quadrature mode is used, set SW6 and SW7 to the proper multiplier (refer to Appendix A).

4) Use jumper (J1) to either enable or disable the optoisolation (see Figure 2.1.1).

5) Insert the OptiTracker card into a standard 8 bit or 16 bit slot (preferably the second or third slot from the power supply).

6) Tighten the retaining screw.

7) Turn on the power.

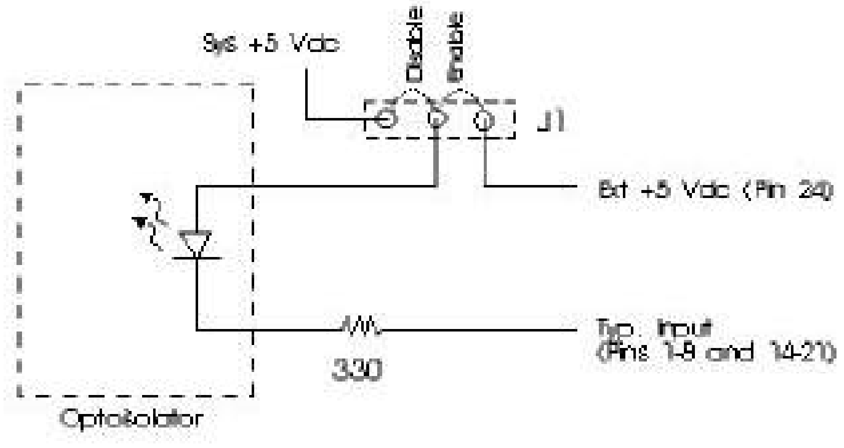


Figure 2.1.1 OptiTracker Typical Input Diagram

2.2 Files on Disk

The OptiTracker support software is supplied on a 3-1/2" High Density floppy disk. The disk contain two demonstration programs and support routines that can be used in your own programs. The support routines are compatible with Quick-C, version 2.5 or later; Microsoft C, version 6.0 or later; BASIC PDS, version 7.0 or later; and QuickBASIC, version 4.5 or later. These routines are supplied in a C library in both small and medium models as well as in .LIB and .QLB formats for BASIC. A sample program for both languages is included and can be used, as is, modified, or portions may be included, in your own programs.

README	.TXT	- contains new information not in the current manual
DRO	.EXE	- ready-to-use digital readout program
DRO	.C	- source code for the DRO program (C language demo)
DRO	.DEF	- address definition file (required for the DRO program)
HELVB	.FON	- helvetica font file (required for the DRO program)
OTDEMO	.EXE	- ready-to-use demonstration program
OTDEMO	.BAS	- source code for the BASIC demo
OT_CS	.LIB	- C small model library
OT_CM	.LIB	- C medium model library
OT_CL	.LIB	- C Large model library
OT_C	.H	- C header file
OT_CPP	.H	- C++ header file
OT_BC	.LIB	- Basic 7 library (compiler)
OT_BC	.QLB	- Basic 7 library (for the environment)
OT_BC	.BI	- Basic include file
OT_QB	.LIB	- Basic 4.5 library (compiler)
OT_QB	.QLB	- Basic 4.5 library (for the environment)

2.3 Running the Demo Programs

BASIC Demo

If you have installed the OptiTracker card in the computer, you can immediately start counting pulses by running the executable version (OTDEMO.EXE), or you can use the source code and run the demo directly from the environment.

To get started in BASIC right away, simply start QuickBASIC 4.5 by loading the OT_QB.QLB Quick library by typing this at the DOS prompt:

```
QB /L OT_QB
```

or for BASIC 7, type:

```
QBX /L OT_BC
```

Now you may load and run the demo program OTDEMO.BAS.

C Demo

If you have installed the OptiTracker card in the computer and have connected the encoders to the rear connector, you can immediately start monitoring your position data by running the executable version (DRO.EXE), or you can use the source code and run the demo from the environment, or create your own executable program.

To get started in C right away, simply start QuickC 2.5 by typing this at the DOS prompt:

```
QC
```

Now create a program list (.MAK file) that includes the DRO.C source code and either OT_CS.LIB for a small model program or OT_CM.LIB for a medium model program. Press F5 to run the program.

2.4 Writing Your Own Programs

By adding the OptiTracker low level routines to your high level language, you can write powerful motion tracking applications without the need to learn or program in tedious assembly language.

Follow the guidelines below for the language of your choice.

Quick C or Professional C

Specify a .MAK file that includes the name of the appropriate support library file. Use OT_CM.LIB for medium model or OT_CS.LIB for small model programs. Use the appropriate header file, OT_C.H (for C)or OT_CPP.H (for C++), at the top of your programs to declare the functions found in the library files.

As an alternative to using the header file, cut and paste the functions you will use in your program from the appropriate model header file.

QuickBASIC or Professional BASIC

Be sure to add the line,

```
'$include:'OT_BC.BI
```

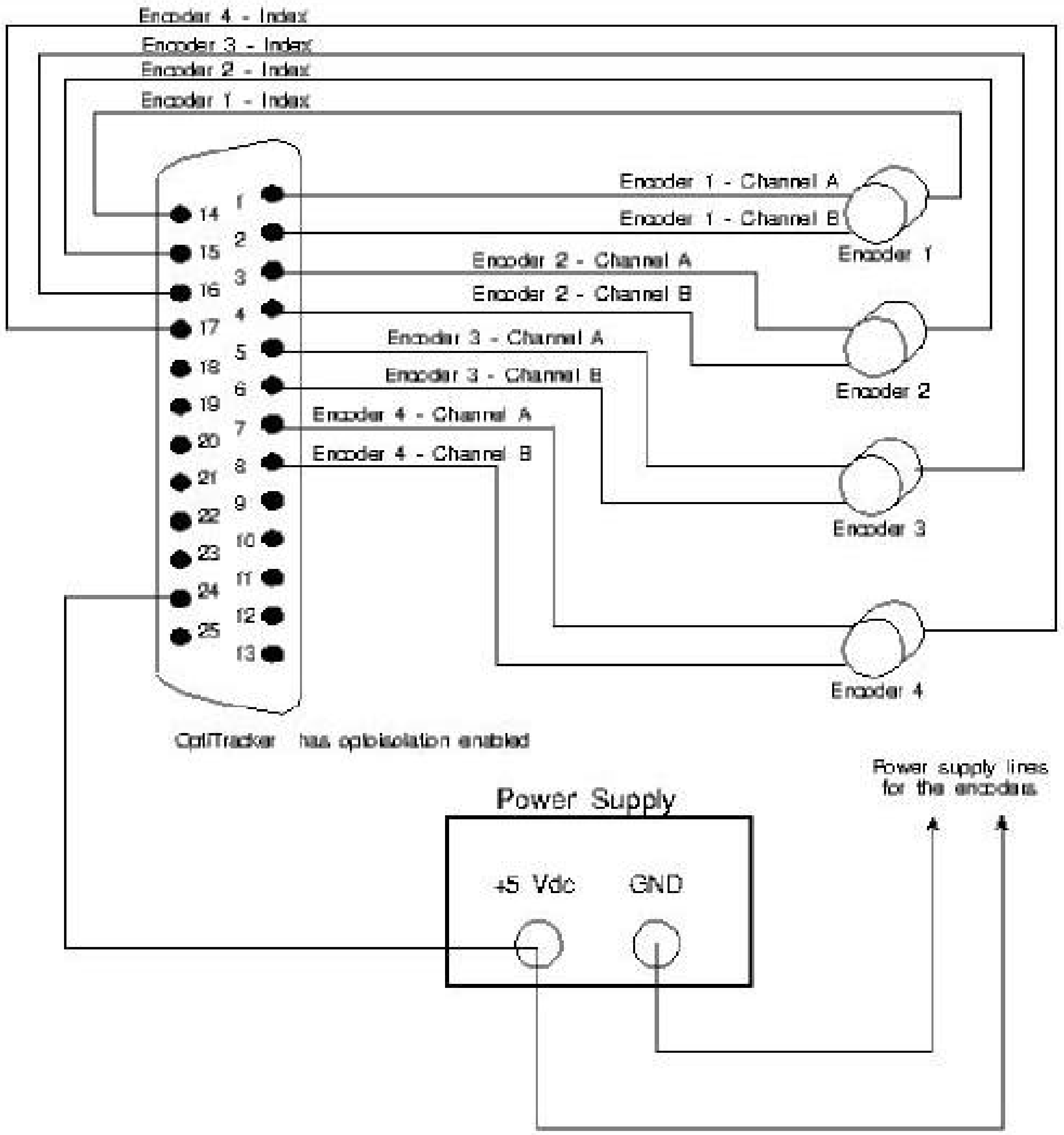
near the beginning of your program. Copy the OT_BC.BI file from the included disk into the directory where BASIC is set to look for include files. Enter BASIC by loading the appropriate Quick library like this:

For Quick BASIC 4.5, type: `QB /L OT_QB`

and for BASIC 7, type: `QBX /L OT_BC`

2.5 OptiTracker Cable Wiring Diagram

Below is a diagram illustrating how to connect encoders to the OptiTracker card.



2.6 Function Quick Reference

TRCLRPOS	Clears the specified counter (1 - 4) to zero
TRREADPOS	Reads the specified counter (1 - 4)
TRSETPOS	Sets the specified counter (1 - 4) to the long integer value specified (value range: -8,388,608 to 8,388,607)
TRCHK	Checks and confirms the presence of an OptiTracker card in the computer
TRHLD	Enables/disables the counters in COUNT mode
TRINFO	Returns the current support library revision number
TRSTS	Reads the 8 bit auxiliary port
TRSYNC	Allows synchronous timing with any one of the 8 bits of the auxillary port

2.7 Library Function Descriptions

This section describes the support routines supplied with the OptiTracker card. The description follow the format outlined below.

Purpose	Describes the use of the routine
Syntax	Show the proper syntax for calling the routine
Parameters	Describes each parameter used in calling the routine
Example	Shows the use of the routine in a typical code fragment

TRCLRPOS

Purpose: TRCLRPOS clears the specified counter to zero.

Syntax:

```
C      trclrpos ( &counter );
BASIC  CALL trclrpos ( counter% )
```

Parameters: The variable **counter** is a short integer which specifies the counter to be cleared. The value range is from 1 to 4 only. Any other value will produce unpredictable results.

Examples:

C

```
unsigned int counter;

printf("Enter counter to be cleared => ");
scanf("%i",&counter);
trclrpos(&counter);
```

BASIC

```
INPUT "Enter counter to be cleared => ",counter%
CALL trclrpos(counter%)
```

TRREADPOS

Purpose: TRREADPOS reads the value stored in the specified counter.

Syntax:

```
C      trreadpos ( &counter, &value );
BASIC  CALL trreadpos ( counter%, value& )
```

Parameters: The variable **counter** is a short integer which specifies the counter to be read. The value range is from 1 to 4 only. Any other value will produce unpredictable results. **value** is a long signed integer that is returned by the TRREADPOS routine. The range for variable **value** is from -8,388,608 to 8,388,607.

Examples:

C

```
unsigned int  counter;
signed long   value;

printf("Enter counter to be read => ");
scanf("%i",&counter);
trreadpos(&counter,&value);
```

BASIC

```
INPUT "Enter counter to be read => ",counter%
CALL trreadpos(counter%, value&)
```

TRSETPOS

Purpose: TRSETPOS sets the specified counter to the specified value.

Syntax:

```
C      trsetpos ( &counter, &value );
BASIC  CALL trsetpos ( counter%, value& )
```

Parameters: The variable **counter** is a short integer which specifies the counter to be set. The value range is from 1 to 4 only. Any other value will produce unpredictable results. **value** is a long signed integer that is passed by the TRSETPOS routine. The range for variable **value** is from -8,388,608 to 8,388,607.

Examples:

C

```
unsigned int  counter;
signed long   value;

printf("Enter counter to be set => ");
scanf("%i",&counter);
printf("Enter the value => ");
scanf("%li",&value);
trsetpos(&counter,&value);
```

BASIC

```
INPUT "Enter counter to be read => ",counter%
INPUT "Enter the value => ",value&
CALL trsetpos(counter%, value&)
```

TRCHK

Purpose: TRCHK checks and confirms the presence of an OptiTracker card in the computer.

Syntax:

```
C      trchk ( &baseaddr, &retcode );
BASIC  CALL trchk ( baseaddr%, retcode% )
```

Parameters: The variable **baseaddr** is a short integer which contains the baseaddress of the OptiTracker card. This should match the address value selected by the dip switches on the card. If the card is present, variable **retcode** will be 1 upon return otherwise it will be 0.

Examples:

C

```
unsigned int  baseaddr, retcode;

printf("Enter the baseaddress of the OptiTracker card => ");
scanf("%i",&baseaddr);
trchk(&baseaddr,&retcode);
if(retcode == 0) {
   printf("\nOptiTracker card not found at address %d.",baseaddr);
   exit(1);
   }
```

BASIC

```
INPUT "Enter the baseaddress of the OptiTracker card => ",baseaddr%
CALL trchk(baseaddr%,retcode%)
IF retcode% = 0 THEN
  PRINT "OptiTracker card not found at the specified address."
  END

END IF
```

TRHLD

Purpose: TRHLD enables/disables the specified counters in COUNT mode.

Syntax:

C `trhld ( &holdbyte );`
BASIC `CALL trhld ( holdbyte% )`

Parameters: The variable **holdbyte** is a short integer which specifies which set of counters are enabled or disabled. The value range is from 0 to 3. The control codes are as follows: 0 - disables all four counters
1 - enables counters 1 and 3
2 - enables counters 2 and 4
3 - enables all four counters
NOTE: All four counters are enabled at power-up.

Only the least significant 2 bits of word **holdbyte** are used therefore any valid short integer value will be accepted (Example: HOLDBYTE = 2 will do the same thing as HOLDBYTE = 250).

Examples:

C

```
unsigned int  holdbyte1=0, holdbyte2=1, counter = 1;
signed long   value;
time_t        cstart1, cstart2, cend;
float         frequency, time;

trhld(&holdbyte1);  /* Disable counter 1 */
trclrpos(&counter);
do
  cstart1 = clock();
while((cstart2 = clock()) == cstart1);
trhld(&holdbyte2);  /* Enable counter 1 */

/* Set delay loop here */

cend = clock();
trhld(&holdbyte1);  /* Disable counter 1 */
trreadpos(&counter, &value);
time = (float) (cend-cstart) / CLK_TCK;
frequency = (float) value / time;
```

BASIC

```
PRINT "Enter '0' to disable all counters"
PRINT "Enter '1' to enable counters 1 and 3"
PRINT "Enter '2' to enable counters 2 and 4"
INPUT "Enter '3' to enable all counters", holdbyte%
CALL trhld(holdbyte%);
```

TRINFO

Purpose: TRINFO returns the current support library revision number.

Syntax:

C `trinfo ( &rev );`

BASIC `CALL trinfo ( rev% )`

Parameters: The variable **rev** is a short integer which contains the revision number of the current support routine library times 10. To find the current revision, simply call TRINFO and divide variable **rev** by 10.

Examples:

C

```
unsigned int  rev;
float         revision;

trinfo(&rev);
revision = (float) rev / 10.0;
```

BASIC

```
CALL trinfo(rev%)
revision! = rev% / 10
```

TRSTS

Purpose: TRSTS returns an 8-bit word from the auxiliary input port.

Syntax:

```
C       trsts ( &status );
BASIC   CALL trsts ( status% )
```

Parameters: The variable **status** is a short integer which contains the data returned by the routine. The port is 8 bits wide and are placed in the least significant 8 bits of the variable **status**.

Examples:

C

```
unsigned int status;

trsts(&status);
if((status & 0xFF) > 0){
  if((status % 2) >= 1) printf("Aux 1 input is high\n");
  if((status % 4) >= 2) printf("Aux 2 input is high\n");
  if((status % 8) >= 4) printf("Aux 3 input is high\n");
  if((status % 16) >=8) printf("Aux 4 input is high\n");
  if((status % 32) >= 16) printf("Aux 5 input is high\n");
  if((status % 64) >= 32) printf("Aux 6 input is high\n");
  if((status % 128) >= 64) printf("Aux 7 input is high\n");
  if((status % 256) >= 128) printf("Aux 8 input is high\n");
   }
else
  printf("All inputs are low\n");
```

BASIC

```
CALL trsts(status%);
IF status% > 0 THEN
  IF status% MOD 2 >= 1 THEN PRINT "Aux 1 input is high"
  IF status% MOD 4 >= 2 THEN PRINT "Aux 2 input is high"
  IF status% MOD 8 >= 4 THEN PRINT "Aux 3 input is high"
  IF status% MOD 16 >= 8 THEN PRINT "Aux 4 input is high"
  IF status% MOD 32 >= 16 THEN PRINT "Aux 5 input is high"
  IF status% MOD 64 >= 32 THEN PRINT "Aux 6 input is high"
  IF status% MOD 128 >= 64 THEN PRINT "Aux 7 input is high"
  IF status% MOD 256 >= 128 THEN PRINT "Aux 8 input is high"
END IF
```

TRSYNC

Purpose: TRSYNC waits until the specified transition of the specified input occurs and then returns to the controlling program.

Syntax:

```
C       trsync ( &bitnum, &polarity );
BASIC   CALL trsync ( bitnum%, polarity% )
```

Parameters: The variable **bitnum** is a short integer which contains the bitnumber to be monitored. The values can range from 1 to 8 only where 1 is the least significant bit. Any other values will produce unpredictable results. **polarity** is a short integer used as a flag with values of 1 or 0 only. If polarity is equal to 1, a low-to-high transition must take place at the input specified by **bitnum** to return. If polarity is 0, a high-to-low transition must take place.

Examples:

C

```
/* Another frequency counter routine
Use a 1 second monostable multivibrator as a count window connected
to pin 14 of the DB25 connector */

unsigned int  bitnum = 1, polarity = 1, counter = 1;
signed long   value;

/* Disable counter 1 here (see TRHLD) */
/* Clear counter 1 here (see TRCLRPOS) */

trsync(&bitnum, &polarity);

/* Enable counter 1 here (see TRHLD) */

polarity=0;
trsync(&bitnum, &polarity);

/* Disable counter 1 here (see TRHLD) */
trreadpos(&counter, &value);

/* The input frequency would equal value */;
```

BASIC

```
inputsync:
INPUT "Enter input on which to sync. (1-8)",bitnum%
IF bitnum% < 1 OR bitnum% > 8 THEN GOTO inputsync
getpolarity:
INPUT "Enter polarity of sync signal (1 = pos, 0 = neg)",polarity%
IF polarity% < 1 OR polarity% > 8 THEN GOTO getpolarity
CALL trsync(bitnum%, polarity%)
```

APPENDIX A

Dip Switch Settings

Address Settings (SW1 - SW5):

SW1	SW2	SW3	SW4	SW5	Hex	Dec
ON	ON	ON	ON	ON	200	512
ON	ON	ON	ON	OFF	210	528
ON	ON	ON	OFF	ON	220	544
ON	ON	ON	OFF	OFF	230	560
ON	ON	OFF	ON	ON	240	576
ON	ON	OFF	ON	OFF	250	592
ON	ON	OFF	OFF	ON	260	608
ON	ON	OFF	OFF	OFF	270	624
ON	***OFF***	***ON***	***ON***	***ON***	***280***	***640****
ON	OFF	ON	ON	OFF	290	656
ON	OFF	ON	OFF	ON	2A0	672
ON	OFF	ON	OFF	OFF	2B0	688
ON	OFF	OFF	ON	ON	2C0	704
ON	OFF	OFF	ON	OFF	2D0	720
ON	OFF	OFF	OFF	ON	200	736
ON	OFF	OFF	OFF	OFF	2F0	752
OFF	ON	ON	ON	ON	300	768
OFF	ON	ON	ON	OFF	310	784
OFF	ON	ON	OFF	ON	320	800
OFF	ON	ON	OFF	OFF	330	816
OFF	ON	OFF	ON	ON	340	832
OFF	ON	OFF	ON	OFF	350	848
OFF	ON	OFF	OFF	ON	360	864
OFF	ON	OFF	OFF	OFF	370	880
OFF	OFF	ON	ON	ON	380	896
OFF	OFF	ON	ON	OFF	390	912
OFF	OFF	ON	OFF	ON	3A0	928
OFF	OFF	ON	OFF	OFF	3B0	944
OFF	OFF	OFF	ON	ON	3C0	960
OFF	OFF	OFF	ON	OFF	3D0	976
OFF	OFF	OFF	OFF	ON	300	992
OFF	OFF	OFF	OFF	OFF	3F0	1008

* Default Setting

Function Settings (SW6 - SW8):

SW6	SW7	SW8	Function
OFF	OFF	OFF	Quadrature Mode - 4x (default)
OFF	ON	OFF	Quadrature Mode - 2x
ON	OFF	OFF	Quadrature Mode - 3x
ON	ON	OFF	Quadrature Mode - 1x
---	---	ON	Count Mode

APPENDIX B

Avoiding I/O address conflicts

In setting the address dip switches on the OptiTracker card, you must choose an address that is not in conflict with another card in your system. The following table lists I/O port addresses used by standard devices in the IBM PC. Select an address that avoids the ones used in the table below as well as avoiding any addresses used by any special hardware that might be in your system such as tape backup hardware, video digitizers, scanners, networking cards etc..

I/O Address	I/O Channel
1F0-1F8	AT fixed disk
200-20F	Game I/O adapter
210-217	Expansion unit
220-24F	Reserved
250-277	Not used
278-27F	Second parallel printer interface (LPT2)
280-2EF	Not used
2F0-2F7	Reserved
2F8-2FF	Second 8250 serial UART interface (COM2)
300-31F	Prototype card
320-32F	XT hard disk
330-377	Not used
378-37F	First parallel printer interface (LPT1)
380-38C	SDLC or secondary binary synchronous interface
390-39F	Not used
3A0-3AF	Primary binary synchronous
3B0-3BF	Monochrome display and first parallel printer
3C0-3CF	Reserved
3D0-3DF	Color/graphics display adaptor
3E0-3EF	Reserved
3F0-3F7	5-1/4 floppy disk drive controller
3F8-3FF	First 8250 serial UART interface (COM1)

APPENDIX C

DB25 CONNECTOR PIN DESCRIPTIONS

PIN#	DB-25 DESCRIPTION	Quadrature Mode	Count Mode
1	Counter 1 Input	Channel A	Direction
2	Counter 1 Input	Channel B	Pulse
3	Counter 2 Input	Channel A	Direction
4	Counter 2 Input	Channel B	Pulse
5	Counter 3 Input	Channel A	Direction
6	Counter 3 Input	Channel B	Pulse
7	Counter 4 Input	Channel A	Direction
8	Counter 4 Input	Channel B	Pulse
9	N/C		
10	N/C		
11	N/C		
12	N/C		
13	N/C		
14	Auxiliary Input 1		
15	Auxiliary Input 2		
16	Auxiliary Input 3		
17	Auxiliary Input 4		
18	Auxiliary Input 5		
19	Auxiliary Input 6		
20	Auxiliary Input 7		
21	Auxiliary Input 8		
22	N/C		
23	System Ground		
24	External +5 Vdc		
25	System +5 Vdc		

APPENDIX D

DRO.C Program Listing

```
/*
      Note:  Specify a .MAK file which includes the OT_CS.LIB
             for small model programs or OT_CM.LIB for medium
             model programs.

Study and use this this code for reference.
Feel free to copy and paste any portion you need into your program.

This demostration program turns any IBM-PC into a digital readout (DRO)
for monitoring position up to four axes.  You must have minimum EGA
graphics and an OptiTracker card in the PC.

TRCHK checks for the presence of the OptiTracker card at the address
specified by baseaddr.  An OptiTracker card was not found if retcode equals
zero.

TRREADPOS reads the position counter of the axis specified by variable axis.
Legal values of variable axis are 1, 2, 3, 4.  The value read is a long
integer between -8,388,608 and 8,388,607 and stored at the location pointed
to by variable count.

TRCLRPOS clears the position counter of the axis specified by variable axis.
Legal values of variable axis are 1, 2, 3, 4.

TRSETPOS sets the position counter of the axis specified by variable axis.
Legal values of variable axis are 1, 2, 3, 4.  The value set is a long
integer between -8,388,608 and 8,388,607 and stored at the location pointed
to by variable count.

                                                */


/* Include files */
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <graph.h>
#include <malloc.h>
#include <bios.h>

/* Define Statements */
#define COLORS 10          /* Use high-intensity colors */
#define PRECISION 4        /* Precision to 4 decimal places */
#define FILEPARM 5         /* Number of parameters in the .DEF file */
#define peek(addr)  (*(unsigned char _far *)addr)
```

```
/* Prototypes */
void set_screen(struct xycoord *,struct xycoord *,\
            struct xycoord *,struct xycoord *,\
            struct xycoord *);
void write_position(struct xycoord *, char *, int, char *);
void make_float_string(float, char *, int );
void menu_reset(int *, char *);
void safe_exit(char *);
void get_val(long *, unsigned long *,struct xycoord *, char  *);

/* Functions found in library OP_CS.LIB  and OP_CM.LIB */
extern void trchk(int *, int *);
extern void trreadpos(int *, long *);
extern void trclrpos(int *);
extern void trsetpos(int *, long *);

/* Global Variables */
char x_axis_position[50] = {0},
     y_axis_position[50] = {0},
     z_axis_position[50] = {0},
     w_axis_position[50] = {0};
unsigned char  *list[2] = {“t’Helv’h48b”,”t’Helv’h14b”};


struct videoconfig vc;         /* Video Information */
struct _fontinfo   fis,fil;    /* Font Information */
int                baseaddress,retcode=0;
unsigned int       kready = _KEYBRD_READY,kread = _KEYBRD_READ;


main()             /* Main */
{

FILE *deffile;
unsigned long   encoder1, encoder2, encoder3, encoder4;
  signed long   counter1, counter2, counter3, counter4,
            prevcnt1=0, prevcnt2=0, prevcnt3=0, prevcnt4=0,
            setnum;
unsigned char   linebuf[FILEPARM][10],menu_key,key[2]= {0};
unsigned int    address, choice, flag=0, halfxpix, cnt=0, boxsize, wait, \
            key_flag=0, COUNTER1=1, COUNTER2=2, COUNTER3=3, COUNTER4=4;
struct xycoord  prevx, prevy, prevz, prevw, valpos;
char            buffer[50],*pixbuff;
float           floattemp;


/* Read .DEF file */
if( (deffile = fopen( “DRO.DEF”, “rt” )) == NULL )
  {
  printf( “Can not open DRO.DEF file.\n” );
  exit( 1 );
  }
```

```
while( !feof( deffile ) )
  {
  if( (fgets( &linebuf[cnt][0], 9, deffile ) == NULL) || cnt >= FILEPARM ) break;
  ++cnt;
  }
fclose( deffile );

/* Assign variables to the .DEF parameters */
baseaddress  = atoi(&linebuf[0][0]);
encoder1 = atol(&linebuf[1][0]);
encoder2 = atol(&linebuf[2][0]);
encoder3 = atol(&linebuf[3][0]);
encoder4 = atol(&linebuf[4][0]);

/* Register fonts */
if( _registerfonts( “*.FON”) <= 0 )
  {
  _outtext( “Error: can’t register fonts” );
  exit( 2 );
  }
if(_setfont(list[0]) < 0)    exit (3); /* Set Large Font   */
if(_getfontinfo( &fil ))     exit (4); /* Get Font information */

/* Set highest available graphics mode and get configuration. */
if( !_setvideomode( _MAXRESMODE ) )
  exit( 5 );
_getvideoconfig( &vc );

/* Create black blanking box */
boxsize = _imagesize(0,0,fil.avgwidth*15,fil.pixheight+8);
pixbuff = malloc( boxsize);
if( pixbuff == NULL ){
   safe_exit(pixbuff);
   printf(“Insufficient Memory”);
   exit( 6 );}
_setcolor(0);
if(_rectangle(_GFILLINTERIOR,0,0,fil.avgwidth*15,fil.pixheight+8) == 0){
   safe_exit(pixbuff);
   printf(“Can’t make graphic”);
   exit( 7 );}
_getimage(0,0,fil.avgwidth*15,fil.pixheight+8,pixbuff);
if ((retcode=_grstatus())<0){
   safe_exit(pixbuff);
   printf(“Unable to get image”);
   exit( 8 );}

/* Output DRO screen */
set_screen(&prevx,&prevz,&prevy,&prevw,&valpos);
```

```
/* Initialize and clear the keyboard buffer */
if( peek( 0x00400096 ) & 0x10 )
 {
 kread = _NKEYBRD_READ;
 kready = _NKEYBRD_READY;
 }

halfxpix=vc.numxpixels/2;

/* Checks that an OptiTracker card is present */
trchk(&baseaddress, &retcode);
if (retcode == 0)
  {
safe_exit(pixbuff);
printf(“OptiTracker card not found at %i.\n\
Change the 1st parameter in the DRO.DEF file.”,baseaddress);
exit(9);
   }

while(key[0] != 27)  /* Loop until <ESC> is pressed */
{
  while(_bios_keybrd(kready)) _bios_keybrd( kread );  /* Clear the keyboard*/
  while(!(menu_key=(char)_bios_keybrd(kready)))
    {
    trreadpos(&COUNTER1,&counter1);
    if((prevcnt1 != counter1) || flag == 0)
      {
      make_float_string(((float) counter1 / encoder1),x_axis_position,PRECISION);
      if(flag==0){
      _moveto(halfxpix-_getgtextextent(x_axis_position),\
             prevx.ycoord);
      prevx=_getcurrentposition();
      }
      write_position(&prevx,x_axis_position,0,pixbuff);
      prevcnt1=counter1;
    }

    trreadpos(&COUNTER2,&counter2);
    if((prevcnt2 != counter2) || flag == 0)
      {
      make_float_string(((float) counter2 / encoder2),z_axis_position,PRECISION);
      if(flag==0) {
      _moveto(halfxpix-_getgtextextent(z_axis_position),\
            prevz.ycoord);
      prevz=_getcurrentposition();
      }
      write_position(&prevz,z_axis_position,1,pixbuff);
      prevcnt2=counter2;
    }

    trreadpos(&COUNTER3,&counter3);
    if((prevcnt3 != counter3) || flag == 0)
      {
      make_float_string(((float) counter3 / encoder3),y_axis_position,PRECISION);
      if(flag==0) {
      _moveto(halfxpix-_getgtextextent(y_axis_position),\
            prevy.ycoord);
      prevy=_getcurrentposition();
      }
      write_position(&prevy,y_axis_position,2,pixbuff);
      prevcnt3=counter3;
    }

    trreadpos(&COUNTER4,&counter4);
    if((prevcnt4 != counter4) || flag == 0)
      {
      make_float_string(((float) counter4 / encoder4),w_axis_position,PRECISION);
      if(flag==0) {
      _moveto(halfxpix-_getgtextextent(w_axis_position),\
             prevw.ycoord);
      prevw=_getcurrentposition();
      flag = 1;
       }
      write_position(&prevw,w_axis_position,3,pixbuff);
      prevcnt4=counter4;
    }
    wait = 0;              /* Delay loop to reduce flicker on the screen */
    while (wait < 40000) wait++;
  }

  if(key_flag==1)
    key[1] = menu_key;

  if(key_flag==0)
    {
    key[0] = menu_key;
    key_flag = 1;
    }


  switch(toupper(key[0]))
    {
    case  ‘C’ :  switch(toupper(key[1]))
             {
             case ‘1’: trclrpos(&COUNTER1);
                     menu_reset(&key_flag,key);
                     break;
             case ‘2’: trclrpos(&COUNTER2);
                     menu_reset(&key_flag,key);
                     break;
             case ‘3’: trclrpos(&COUNTER3);
                     menu_reset(&key_flag,key);
                     break;
             case ‘4’: trclrpos(&COUNTER4);
                     menu_reset(&key_flag,key);
                     break;
             case ‘A’: trclrpos(&COUNTER1);
                     trclrpos(&COUNTER2);
                     trclrpos(&COUNTER3);
                     trclrpos(&COUNTER4);
                     menu_reset(&key_flag,key);
                     break;
             case 0  : break;
             default : menu_reset(&key_flag,key);
             }
    break;

    case  ‘S’ :  switch(toupper(key[1]))
             {
             case ‘1’: get_val(&setnum,&encoder1,&valpos,pixbuff);
                     trsetpos(&COUNTER1,&setnum);
                     menu_reset(&key_flag,key);
                     break;
             case ‘2’: get_val(&setnum,&encoder2,&valpos,pixbuff);
                     trsetpos(&COUNTER2,&setnum);
                     menu_reset(&key_flag,key);
                     break;
             case ‘3’: get_val(&setnum,&encoder3,&valpos,pixbuff);
                     trsetpos(&COUNTER3,&setnum);
                     menu_reset(&key_flag,key);
                     break;
             case ‘4’: get_val(&setnum,&encoder4,&valpos,pixbuff);
                     trsetpos(&COUNTER4,&setnum);
                     menu_reset(&key_flag,key);
                     break;
             case 0  : break;
             default : menu_reset(&key_flag,key);
             }

    case  27  : break;
    default   : menu_reset(&key_flag,key);
    }

}
while(_bios_keybrd(kready)) _bios_keybrd( kread );  /* Clear the keyboard*/
safe_exit(pixbuff);
exit(0);
}
/* End of Program */
/* * * * * * * * * * *  Subroutines  * * * * * * * * * * * * * */
/* Sets  up the DRO screen */
void set_screen(struct xycoord *pos1,struct xycoord *pos2,\
            struct xycoord *pos3,struct xycoord *pos4,\
            struct xycoord *pos5)
{
struct xycoord temp;

int xpos = vc.numxpixels/20,ywinmax = 3 * vc.numypixels /4;
int max_width;
unsigned char *axis[4] = {“X-AXIS:”,”Y-AXIS:”,”Z-AXIS:”,”W-AXIS:”};
max_width = _getgtextextent(axis[3]);

_setcolor(COLORS);
_setgtextvector( 1, 0 );
_moveto(xpos+(max_width-_getgtextextent(axis[0])),\
      ywinmax - 4 * ywinmax/5);
_outgtext(axis[0]);
_setcolor(COLORS+1);
temp=_moveto(xpos+(max_width-_getgtextextent(axis[1])),\
            ywinmax - 3 * ywinmax/5);
*pos1=temp;
_outgtext(axis[1]);
_setcolor(COLORS+2);
temp=_moveto(xpos+(max_width-_getgtextextent(axis[2])),\
            ywinmax - 2 * ywinmax/5);
*pos2=temp;
_outgtext(axis[2]);
_setcolor(COLORS+3);
temp=_moveto(xpos,ywinmax - ywinmax/5);
_outgtext(axis[3]);
*pos3=temp;
temp = _getcurrentposition();
*pos4=temp;
_setcolor(COLORS-3);
_rectangle(_GBORDER,0,0,vc.numxpixels-1,vc.numypixels-1);
_rectangle(_GBORDER,xpos-10,pos1->ycoord+fil.pixheight+10,\
                vc.numxpixels-20,pos1->ycoord-10);
_rectangle(_GBORDER,xpos-10,pos2->ycoord+fil.pixheight+10,\
                vc.numxpixels-20,pos2->ycoord-10);
_rectangle(_GBORDER,xpos-10,pos3->ycoord+fil.pixheight+10,\
                vc.numxpixels-20,pos3->ycoord-10);
_rectangle(_GBORDER,xpos-10,pos4->ycoord+fil.pixheight+10,\
                vc.numxpixels-20,pos4->ycoord-10);

if(_setfont(list[1]) < 0)    exit (10);
if(_getfontinfo( &fis ))     exit (11);
_setcolor(COLORS+7);
_moveto(10,ywinmax+fis.pixheight+5);
_outgtext(“Press:”);
_moveto(10,ywinmax+2*(fis.pixheight+5));
_outgtext(“<C> and <1>, <2>, <3>, <4>, or <A>ll to Clear Position(s)”);
_moveto(10,ywinmax+3*(fis.pixheight+5));
_outgtext(“<S> and <1>, <2>, <3>, or <4> to Set Position”);
```

```
temp = _getcurrentposition();
_moveto(10,ywinmax+4*(fis.pixheight+5));
_outgtext("<ESC> to Exit");
temp.xcoord+=20;
*pos5=temp;

if(_setfont(list[0]) < 0)    exit (12);

}
/* Writes the new value to the screen */
void write_position(struct xycoord *pos, char *str, int c, char  *pix)
{
struct xycoord temp=*pos,
             temp2;

_setcolor(COLORS+c);
_moveto(((vc.numxpixels/2)+20)-_getgtextextent(str),temp.ycoord);
temp2 = _getcurrentposition();
_putimage(temp.xcoord-5,temp.ycoord-4, pix,_GPSET);
_moveto(temp2.xcoord,temp2.ycoord);
_outgtext(str);

if ((retcode=_grstatus())<0){
   safe_exit(pix);
   printf("Unable to put image");
   exit( 13 );}
*pos = temp2;
}

/* Creates a string of a floating point number */
void make_float_string(float num, char *str, int p)
{
int decimal, sign;
char *pnumstr,
      tmpbuf[50];

strcpy(str,NULL);  /* Clear the string */

    /* Use information from fcvt to format number string. */
pnumstr = fcvt( num, p, &decimal, &sign );

    /* Start with sign if negative. */
if( sign )
  strcat( str, "-" );

if( decimal <= 0 )
  {
   /* If decimal is to the left of first digit (decimal negative),
    * put in leading zeros, then add digits.
    */
  strcat( str, "0." );
  memset( tmpbuf, '0', (size_t)abs( decimal ) );
  tmpbuf[abs( decimal )] = 0;
  strcat( str, tmpbuf );
  strcat( str, pnumstr );
  }
else
  {
   /* If decimal is to the right of first digit, put in leading
    * digits. Then add decimal and trailing digits.
    */
  strncat( str, pnumstr, (size_t)decimal );
  strcat( str, "." );
  strcat( str, pnumstr + decimal );
  }
}

/* Clears the menu choice array */
void menu_reset(int *flag, char p[])
{
*flag = 0;
p[0]=0;
p[1]=0;
}

/* Exits the program safely.  Releases all fonts and buffers */
void safe_exit(char *buf)
{
_unregisterfonts();
_setvideomode( _DEFAULTMODE );
free(buf);
}

/* Creates a long integer from the user input */
void get_val(long *num, unsigned long *encoder_res,\
           struct xycoord *pos, char  *pix)
{
#define MAXSTRING 10

char buf[MAXSTRING]={0};
struct xycoord temp;
int cnt=0,flag=1;
float value, numtemp;

if(_setfont(list[1]) < 0)    exit (14);
_setcolor(COLORS);
_moveto(pos->xcoord,pos->ycoord);
_outgtext("Value:");
temp = _getcurrentposition();
temp.xcoord+=10;

while(_bios_keybrd(kready)) _bios_keybrd( kread );
while(((buf[cnt]=getch()) != 13) && (cnt < MAXSTRING-1))
  {
  _outgtext(&buf[cnt]);
  ++cnt;
  }
```

```
_putimage(pos->xcoord,pos->ycoord, pix,_GPSET);
if(_setfont(list[0]) < 0)    exit (15);

value=atof(buf);
if(value < 0) flag=-1;
numtemp = value * *encoder_res;

*num = ((long)(numtemp+(0.5*flag)) != (long) numtemp) \
       ? ((long) numtemp)+(1*flag) : (long) numtemp;
}
```

APPENDIX E

OTDEMO.BAS Program Listing

```
‘Study and use this code for reference.
‘Feel free to copy and paste into your program any portion you need.

‘$INCLUDE: ‘ot_bc.bi’

ms1$  = “   —————————————————————————————————————”
ms2$ = “Program:       OptiTracker Demo “
ms3$ = “”
CALL trinfo(rev%)
drvrev$ = STR$(INT(rev% / 10)) + “.” + RIGHT$(STR$(rev%), 1)
ms4$ = “           “ + “Driver version: “ + drvrev$
ms5$ = “(C) Copyright 1993  MicroKinetics Corporation”
ms6$ =     “—————————————————————————————————————”

CLS
PRINT ms1$
PRINT ms2$
PRINT ms3$
PRINT ms4$
PRINT ms5$
PRINT ms6$
PRINT
PRINT
PRINT

start:
INPUT “Enter base address (Normally enter 640):”, baseadress%
CALL trchk(baseadress%, returncode%)
IF returncode% = 0 THEN
PRINT
PRINT “OptiTracker not found in system at specified address...”
PRINT “The address must correspond to the dip switch settings on the card”
PRINT
PRINT “Press any key to resume”
DO WHILE INKEY$ = “”: LOOP
GOTO start
END IF

looper:
PRINT
PRINT “a. Clear all counts”
PRINT “b. Clear one axis”
PRINT “c. Set count”
PRINT “d. Read count”
PRINT “e. Status of inputs”
PRINT “f. Disable counters”
PRINT “g. Sync to an input”
PRINT “<esc> ends demo”
a$ = “”
DO WHILE a$ = “”: a$ = INKEY$: LOOP
a$ = LCASE$(a$): IF a$ = CHR$(27) THEN END
a = ASC(a$) - 96
IF a < 1 OR a > 7 THEN GOTO looper
ON a GOSUB clearall, clearaxis, setnewcount, readcounters, inputstatus, disable,
inputsync
GOTO looper

‘============================================================================
clearall:
FOR axis% = 1 TO 4
CALL trclrpos(axis%)
NEXT axis%
RETURN
‘====================================
clearaxis:
INPUT “Enter axis to clear. (1-4) “, axis%
IF axis% < 1 OR axis% > 4 THEN GOTO clearaxis
CALL trclrpos(axis%)
RETURN
‘====================================
setnewcount:
INPUT “Enter axis to set (1-4) “, axis%
IF axis% < 1 OR axis% > 4 THEN GOTO setnewcount
INPUT “Enter new count: “, count&
CALL trsetpos(axis%, count&)
RETURN
‘====================================
readcounters:
PRINT “M1”, “M2”, “M3”, “M4”
FOR axis% = 1 TO 4
CALL trreadpos(axis%, counter&)
PRINT counter&,
NEXT axis%
RETURN
‘====================================
inputstatus:
DO
CLS
LOCATE 1, 1
CALL trsts(inputbyte%)
PRINT “Input byte in decimal:”; inputbyte%
PRINT “Input byte in hex: “; HEX$(inputbyte%)
```

```
IF inputbyte% > 0 THEN
 IF inputbyte% MOD 2 >= 1 THEN PRINT “Aux input 1 is high” ‘bit 0 high
 IF inputbyte% MOD 4 >= 2 THEN PRINT “Aux input 2 is high” ‘bit 1 high
 IF inputbyte% MOD 8 >= 4 THEN PRINT “Aux input 3 is high” ‘bit 2 high
 IF inputbyte% MOD 16 >= 8 THEN PRINT “Aux input 4 is high” ‘bit 3 high
 IF inputbyte% MOD 32 >= 16 THEN PRINT “Aux input 5 is high” ‘bit 4 high
 IF inputbyte% MOD 64 >= 32 THEN PRINT “Aux input 6 is high” ‘bit 5 high
 IF inputbyte% MOD 128 >= 64 THEN PRINT “Aux input 7 is high” ‘bit 6 high
 IF inputbyte% MOD 256 >= 128 THEN PRINT “Aux input 8 is high” ‘bit 7 high
END IF

FOR x = 1 TO 1000: NEXT
LOOP WHILE INKEY$ = “”
RETURN
‘====================================
disable:
PRINT “Enter ‘0’ to disable all counters.”
PRINT “Enter ‘1’ to enable counters M1 and M3.”
PRINT “Enter ‘2’ to enable counters M2 and M4.”
INPUT “Enter ‘3’ to enable all counters. “, hold%
CALL trhld(hold%)
RETURN
‘====================================
inputsync:
INPUT “Enter input on which to sync. (1-8)”, inputbit%
IF inputbit% < 1 OR inputbit% > 8 THEN GOTO inputsync
getpolarity:
INPUT “Enter polarity of sync signal (1 = positive, 0 = negative) “, polarity%
IF polarity% < 0 OR polarity% > 1 THEN GOTO getpolarity

CALL trsync(inputbit%, polarity%)
RETURN
‘======================================
```