TRIGGER SYSTEM FOR MULTIPLE MIRROR EXPERIMENT

by

B. T. Archer, R. T. Hamilton, and H. Meuth

Memorandum No. UCB/ERL M85/80

4 October 1985

(COVER PAGE)

TRIGGER SYSTEM FOR MULTIPLE MIRROR EXPERIMENT

by

B. T. Archer, R. T. Hamilton, and H. Meuth

Memorandum No. UCB/ERL M85/80

4 October 1985

Title page

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TRIGGER SYSTEM FOR MULTIPLE MIRROR EXPERIMENT

B. T. Archer, R. T. Hamilton, and H. Meuth

Contents

I. Overview

The trigger system for the Berkeley Ten Meter Multiple Mirror Experiment (MMX) is illustrated in Fig. 1. The Main Timing Unit is located in the MMX screen room. This unit contains a trigger generator circuit, 32 computer-programmable time delay circuits, and 32 optical couplers which link to the MMX control panel. On the MMX control panel reside 32 receivers for the optical coupler circuits which convert light pulses into 120V pulses to trigger the ignitron firing trigger generators.

time delay checking code.

II. Trigger Generator

The trigger generator (Fig. 4) provides trigger inputs to the time delays. When the trigger source is set to automatic, the rate switch permits selection either 1 or 10 triggers per second. When the trigger source is set to manual, a trigger is produced either by the pushbutton switch mounted on the panel, a pushbutton switch connected to the UHF jack on the time delay panel, or by the charging timer for the MMX capacitor banks. The charging timer is the trigger source used during normal operation of the machine; other sources are for equipment testing and debugging purposes. A 10 μsec TTL pulse is produced by the one-shot trigger circuit, and is buffered through a hex buffer to trigger all 32 time delays simultaneously. A LEMO output is

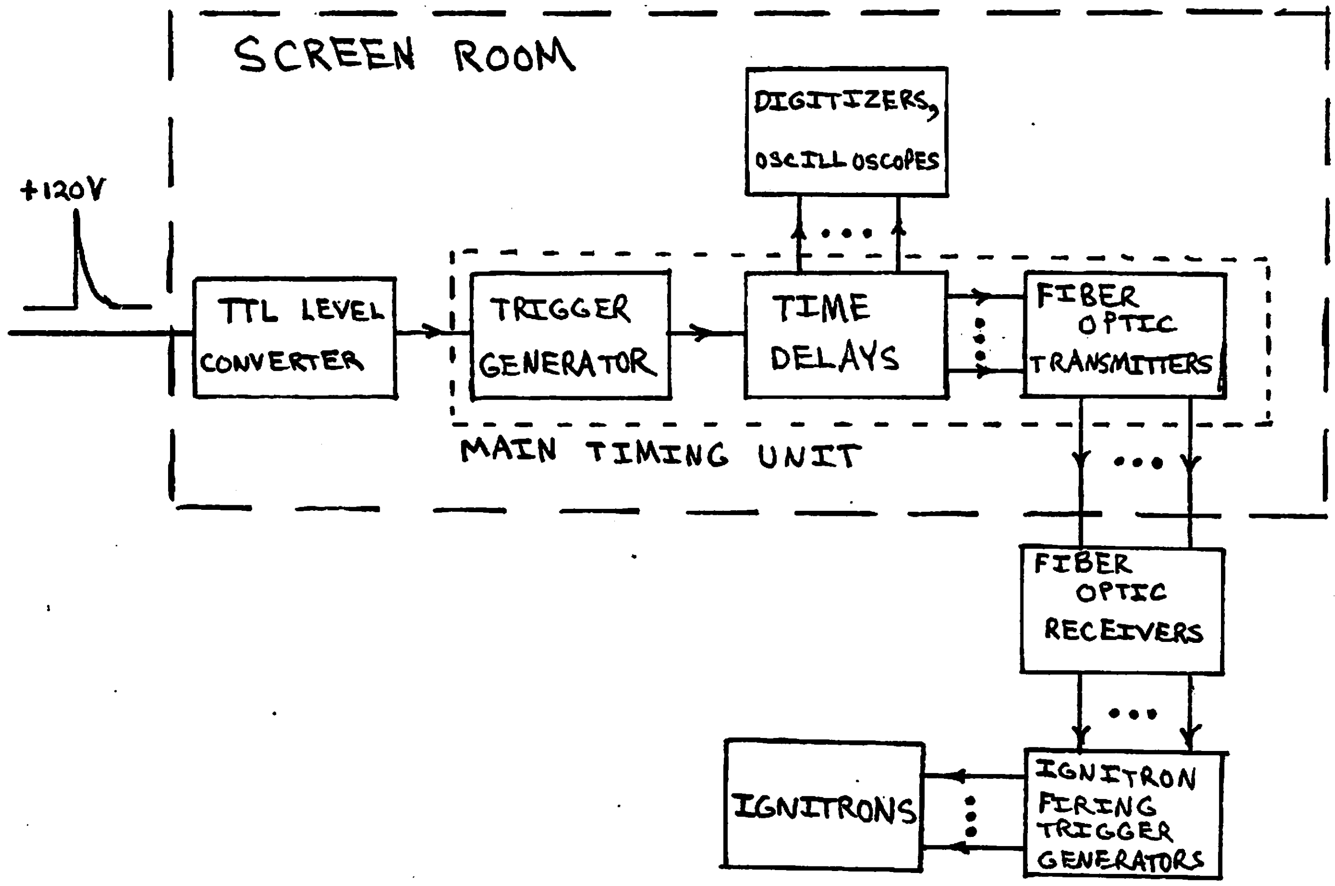


Figure 1: Trigger system block diagram

Appendix A contains circuit diagrams for these components. Appendix B gives the source code listing for TD.C, the time delay programming code. Appendix C gives the source code listing for TDCHECK.C, the automated time delay checking code.

II. Trigger Generator

The trigger generator (Fig. 4) provides trigger inputs to the time delays. When the trigger source is set to automatic, the rate switch permits selection either 1 or 10 triggers per second. When the trigger source is set to manual, a trigger is produced either by the pushbutton switch mounted on the panel, a pushbutton switch connected to the UHF jack on the time delay panel, or by the charging timer for the MMX capacitor banks. The charging timer is the trigger source used during normal operation of the machine; other sources are for equipment testing and debugging purposes. A 10 μsec TTL pulse is produced by the one-shot trigger circuit, and is buffered through a hex buffer to trigger all 32 time delays simultaneously. A LEMO output is provided to monitor the trigger pulse, if need be.

III. Time delay interface board

The time delay interface board (Fig. 5) inserts into the motherboard of the S-100 computer. A 7805 voltage regulator converts the computer's unregulated +8V supply into a regulated +5V supply. A 10MHz clock oscillator provides counting pulses to the time delays through the 34-line ribbon cable. Data for the time delays (DO0–DO7) is buffered through a 74LS245 octal bus tranceiver. The remainder of the board is devoted to selecting which time delay is to be programmed. The 74HCT688 is an 8-bit comparator which provides an low active signal when the lowest 8-bits of the address bus (A0–A7) are equal to the 8-bits of the addressing switch (SW1). The switch is presently set to hex address D0. This means that subswitches A4, A6, and A7 are set to the OPEN position (logical '1'), and subswitches A0, A1, A2, A3, and A5 are set to the CLOSED position (logical '0'). The condition for accessing the time delays is produced when the '688 produces a $\overline{P=Q}$ pulse, the bus signal pWR* goes low, sOUT goes high, and A14 and A15 go low. At this point, the 74LS30 8-input NAND gate produces the low-active VC* signal. When VC* is active the bus address pins A11–A13 select a time delay board. These three lines form a 3-bit value, and can hence select one of $2^3 = 8$ time delay boards. These three lines are buffered through a 74LS245 8-bit octal bus tranceiver to become board select signals BS0–BS2. Bus address lines A8–A10 are also buffered through the '245 bus tranceiver, and these three lines become chip select signals CS0–CS2. This 3-bit value selects one of the eight 8-bit latches on board the time delay board selected by BS0–BS2.

IV. Time Delays

The time delays each have four outputs, each of which is capable of driving an optical coupler, or providing a trigger to a digitizer. Each time delay may be programmed for a delay of 0.1–6553.5 μsec. A time delay may be disabled by setting its delay value to 0. The output of each delay consists of a 40 msec TTL high pulse. The delays are numbered on the front panel from 1 to 32 for convenient reference. Electrically, however, there are 8 time delay boards numbered 0 through 7, and each board contains 4 time delays numbered 0 through 3.

IV.1 Board construction

There are at present 8 completed time delay boards, plus 1 fully tested spare and 2 pre-drilled circuit boards. Construction and testing of a pre-drilled circuit board can be completed in about 2 hours. The circuit boards were fabricated and drilled by Teltek Corp., 1509 Berger Dr., San Jose. The following components are required to assemble each board:

QTY.	Item	Description
Integrated Circuits		
8	74HCT40103	8-bit down counter
8	74HCT373	8-bit latch
2	74LS74	Dual D-type flip-flop
2	74LS123	Dual monostable multivibrator
2	74LS32	Quad OR gate
1	74LS14	Hex Schmitt-trigger inverter
1	74LS245	Octal bus tranceiver
1	74LS175	Quad D-type flip-flop
1	74HCT138	Inverting 3-to-8 line decoder
1	74HCT238	Non-inverting 3-to-8 line decoder
Miscellaneous items		
9		16-pin IC decoupling capacitor
1		14-pin IC decoupling capacitor
4		330Ω 1/4 watt resistors
4		100K 1/4 watt resistors
4		1μF tantalum capacitors
1		34-pin right-angle soldertail header

The decoupling capacitors are used on the 74HCT40103's, the 74LS175, and the 74LS14. The boards are hard-wired as Board 0 through Board 7. Board 0 contains time delays 1–4, Board 1 contains time delays 5–8, etc.

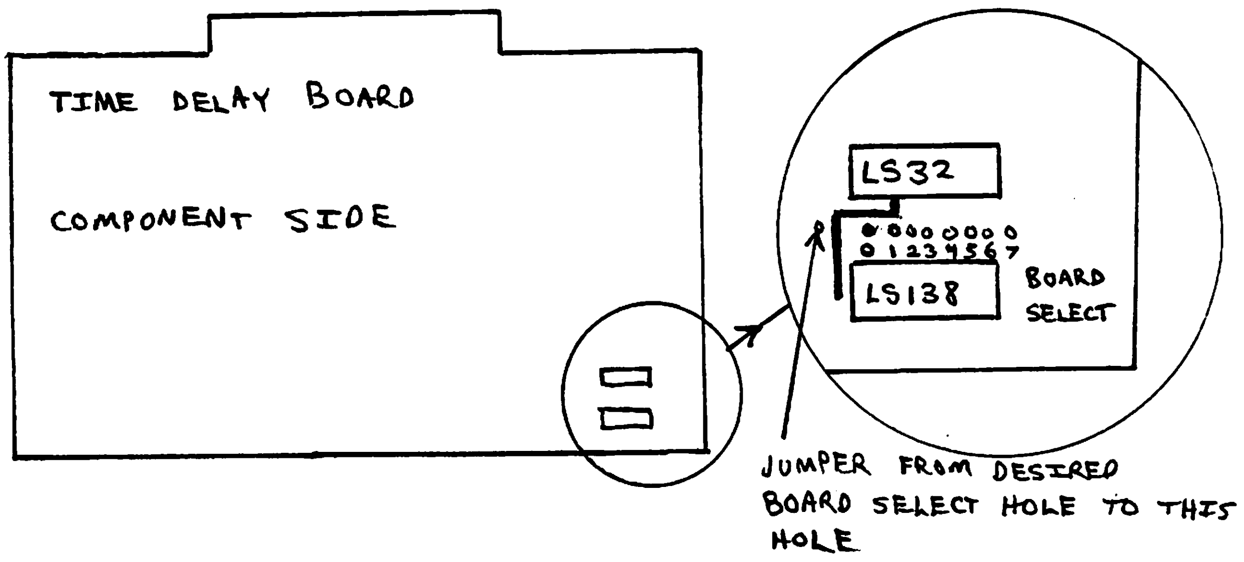


Figure 2: Wiring the board select.

The boards are hard-wired as indicated in Fig. 2. A wire jumper is used to connect one of the outputs of the '138 3-to-8 decoder to pin 2 of a '32 quad OR gate.

Table 1 gives the pin connections for the 44-pin connectors mounted on the time delay panel. The bottom pin on the component side is pin 1. If the panel requires modification, it is import to to minimize the excess wiring on the panel, since the panel is already crowded.

IV.2 Circuit operation

The heart of each time delay circuit (see Fig. 6) is formed by two '40103 8-bit down counters. These two counters are cascaded to form a 16-bit counter. Each pair of counters is wired to two '373 8-bit octal latches which hold the time delay value. The 16-bit down counter is triggered by the input trigger, and counting pulses are provided by the 10MHz signal on board the time delay interface board. When the down counter reaches zero, a 40 msec output pulse is generated which appears on the 4 LEMO outputs on the front of the time delay panel and also resets the counters to the preloaded

Pin	Connection
1	LED 3
2	OUTPUT 3
3	LED 2
4	OUTPUT 2
5	INPUT 3
6	INPUT 2
10	+5V
14	GND
17	LED 0
18	OUTPUT 0
19	LED 1
20	OUTPUT 1
21	INPUT 0
22	INPUT 1

Table 1: Pin connections for time delay modules

count. This output pulse is accompanied by illumination of a red LED which indicates that the delay fired. Since a 16-bit counter can count to a maximum value of $2^{16} - 1 = 65535$, and one count is made each 0.1 μsec, the maximum time delay value is 6.5535 msec.

The time delay latches are programmed by loading data into the '373 octal latches via the 34-line ribbon cable from the computer. The two latches for each time delay are programmed one at a time via the 8-bit data bus. When the board select signals activate the '138 output which is wired into the '32 OR gate input (see Fig. 2), and the VC* signal is active, then the board is selected. The chip select signal (CS0–CS2) selects one of the 8-bit latches to be programmed. For example, to access the low 8-bits of Counter 1 on Board 3, the board select signal is {BS2, BS1, BS0} = {0, 1, 1} and the counter select signal is {CS2, CS1, CS0} = {0, 0, 1}. This combination activates pin 12 ($\overline{Y3}$) of the '138, and pin 14 (Y1) of the '238. The output of the '238 drives down the $\overline{LE}$ pin of the lowermost latch of Fig. 6, causing that latch to accept data from the data bus, which is buffered through a '245 octal bus tranceiver.

The time delay operation is initiated by a high pulse into the '175 quad latch. This signal is synchronized with the 10MHz clock signal to prevent timing problems within the '40103 down counters. The output of the '175 is used to clock high data to the output of a '74 D-type flip-flop. This high output is fed into the $\overline{PL}$ inputs of the '40103 down counters and enables counting. The 'LOW' counter counts at 10MHz, and each time it reaches zero, it sends a clock pulse to the 'HIGH' counter. When both counters reach zero simulteously, the output of a '32 OR gate goes low and initiates an output pulse by the '123 monostable multivibrator. The '32 OR gate output also provides a clear data ($\overline{CD}$) signal to the '74 D-type flip flop. This action brings the output of the flip-flop low, which proceeds to reset the '40103 counters to their preset value, and inhibit counting until the next trigger input.

Note that programming a zero into both latches of a given time delay forces the output of the '32 OR gate to remain low at all times. Normally a down transition on this signal causes the '123 monostable multivibrators to generate an output pulse. When the input is always held low, however, no down transition can be produced, and hence no output pulse can be generated. Therefore the time delays have the useful feature that they can be disabled by setting the delay value to zero.

IV.3 Time delay reliability

The reliability of a given time delay can be checked automatically using the transient digitizers in the MMX screen room. The time delay and digitizer are connected as indicated in Fig. 3. The digitizer offset is adjusted to digitize a positive unipolar signal (e. g. the offset of a Transiac 2008 digitizer is set to −256 mV. The test is automated by running the TDCHECK code, a listing of which is provided in Appendix C. This code is invoked using the following command line:

```
tdcheck <haltflag> <file name>
```

TDCHECK prompts for a digitizer to be checked (1–32), and a digitizer slot to be used. Any type of digitizer can be used (Transiac 2008, Transiac 2001, or LeCroy 8837F). Then TDCHECK prompts for a starting value and a skip value for the time delays. These values are entered as a number of counts (1–65535). The code begins with the given starting value and programs the indicated time delay with that value. The digitizer is then automatically set to an appropriate sampling rate and triggered. After a stop trigger is produced by the trigger generator, TDCHECK examines the digitizer data to compute a measured delay value. If the delay value is within the error limits imposed by the resolution of the digitizer, then the programmed setting and the measured setting are displayed on the S-100 computer console. If there is an error, an error message is written to the output file `<file name>` which was named on the invocation line. If the file name is omitted, the console is used. Normally errors are directed to the printer by using `PRN:` as the file name. After an error, the program can be directed to halt immediately by typing `HALT` for `<haltflag>`. The digitizer data can then be inspected on the MicroAngelo monitor to help diagnose the problem. Any value for `<haltflag>` other than `HALT` will cause the code to proceed until 100 errors are found.

The code cycles through the delay values, each cycle adding the skip value to the delay value, until the maximum value of 65535 is reached. To check

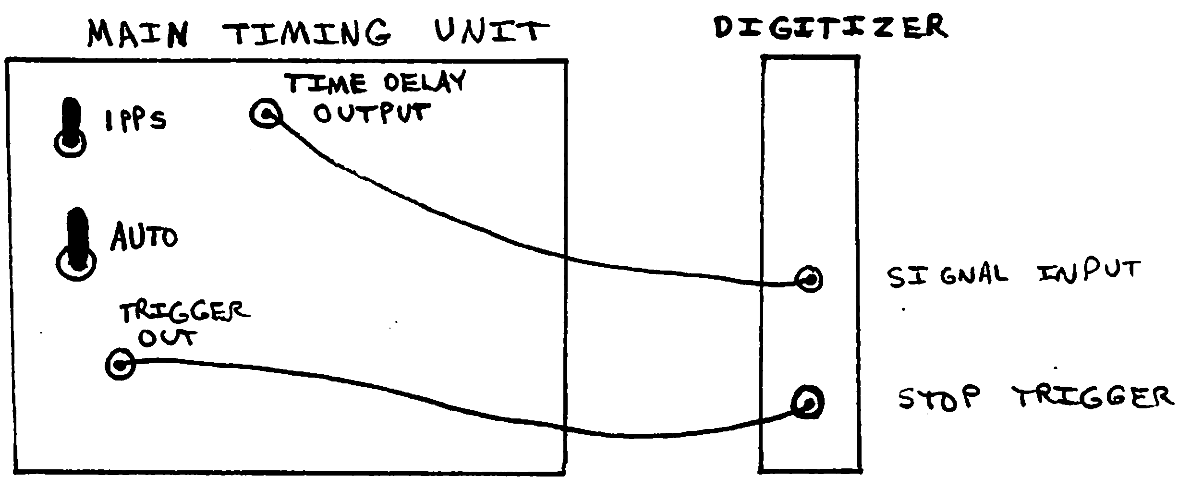


Figure 3: Reliability test for time delays.

every possible time delay value, for example, the starting value is set to 1, and the skip value is set to 1. This complete check requires about 20 hours of running time, and normally zero errors will occur.

A skip value of zero can be used to continually program the same time delay value. This option is useful for debugging purposes. Since the maximum value will never be reached by adding the skip value of zero, the program must by halted by typing Control-C.

IV.4 Time delay programming

Time delay programming is accomplished by the code TD, a listing of which is provided in Appendix B. This code can be reached directly from the main data-taking program DIGIT, or can be invoked by simply typing TD in response to the operating system prompt. TD keeps a file on the local user space called TIMING.DOC. This file keeps a record of the most recently programmed value for each of the 32 time delays, a eight character name for each delay, and a logical trigger (see below) for each delay. This information is also stored at the end of each standard digitizer data file. By typing

```
td <filename>
```

where `<filename>` is a digitizer data file, the time delay settings used in that file are read into memory. Whether the parameters are read from a data file, or from TIMING.DOC, the data is then displayed on the console. To alter the values for a given delay, simply enter the number of the time delay to be changed. For each delay there are three parameters, and each can be retained by simply hitting return when the cursor is located at the appropriate field. Alternatively, the current parameter can be changed by entering the new value, then hitting return.

The logical trigger parameter is a software device for retaining a constant time interval between two time delays. For example, say we wish to trigger the mirror field from time delay 1, and its corresponding crowbar from time delay 2. The logical trigger of delay 1 would be set to 0, so that its value is referenced to the main trigger. But the logical trigger of time delay 2 is set to a value of 1. This means that whenever the value of time delay 1 is changed, the value of time delay 2 will be changed by a similar amount, in order to keep the separation between them constant. It is important to remember that the hardware is actually wired so that all time delays are triggered simultaneously, and the time delay value displayed on the console is always referenced to this trigger.

When modifications are complete to the time delay settings are complete, enter 0 (or simply a carriage return) for the time delay to be modified. This will write the updated file onto TIMING.DOC, and chain back to DIGIT.

V. Optical Couplers

The optical coupler circuit (Fig. 7) provides the connection between the time delays and the Ignitron Firing Chassis. TTL pulses from the time delays are converted to light pulses by fiber optic transmitters in the screen room. A green LED provides a visual indication that the transmitter was triggered. The output of the transmitter is coupled into a fiber optic cable which is connected to a receiver circuit. The receiver circuit consits of a fiber optic receiver which converts the light pulse into a TTL low pulse. A red LED provides visual confirmation that the pulse was received. The TTL low pulse is inverted and triggers a 2N1597 thyristor. The ON current for the thyristor is provided by a .02 μF capacitor which has been charged to +120V between shots through the 47K resistor. When the thyristor is triggered, the capacitor keeps the thyristor on for about 5 μsec. At this point the current through the thyristor drops below the holding current and the device turns off. The current pulse is coupled to the ignitrons through a 1:1 pulse transformer. The risetime of the pulse is $\sim .5\mu$sec.

The total delay between the input trigger to the fiber optic transmitter circuit and the output pulse reaching the ignitron firing threshold of 75 V is about 1.5 μsec.

The fiber optic transmitters and receivers are manufactured by Hewlett-Packard, and are mounted with HFBR4202 mounting hardware. The transmitter part number is HFBR1202, and the receiver part number is HFBR 2202. The fiber optic cable has 200 μm diameter, and is LBL stock catalog number 6145-65673 (Pirelli LED 22214). The fiber optic terminations are manufactured by OFTI (Optical Fiber Technology Inc.), and the part number is 4024 RB-2.7.

The pin numbers used on the fiber optic transmitter circuit are given in Table 2. The first column is the pin number on the 44 pin connector. The next column gives the number of the driver which is connected to that pin, as indicated on the front panel. The first number is for the first transmitter board (drivers 1–15), and the number in parentheses is for the second tranmitter board (drivers 16–32). The third column of the table gives the driver number as labelled on the circuit board, and the fourth column gives the wire color connecting the edge connector pin to either the LED or the LEMO input connector.

The fiber optic cables are terminated according to the techniques outlined in the OFTI guide "Optical Cable Terminating Procedures and Techniques" with the following change: Devcon Five Minute epoxy is used for speed and ease of termination, rather than the slower epoxy which is recommended. The short curing time, however, makes it difficult to get the fiber properly centered in the connector. For this reason the cladding is left on all of the exposed fiber throughout the terminating process. After polishing, this leaves a well centered termination with a finish acceptable for our application.

Pin	Driver #	Board label	wire color
1	15 (31)	INPUT 1	Brown
2	15 (31)	LED 1	Red
3	16 (32)	INPUT 2	Orange
4	16 (32)	LED 2	Yellow
5	13 (28)	INPUT 3	Green
6	13 (28)	LED 3	Blue
7	14 (29)	INPUT 4	Purple
8	14 (29)	LED 4	Gray
10	+5V		
14	GND		
15	4 (19)	INPUT 9	Blue
16	4 (19)	LED 9	Green
17	3 (18)	INPUT 10	Yellow
18	3 (18)	LED 10	Orange
19	1 (16)	INPUT 11	Red
20	1 (16)	LED 11	Brown
21	2 (17)	INPUT 12	Black
22	2 (17)	LED 12	White

Table 2: Pin connections for time delay modules

Pin	Driver #	Board label	wire color
23	12 (27)	INPUT 5	Blue
24	12 (27)	LED 5	Green
25	11 (26)	INPUT 6	Yellow
26	11 (26)	LED 6	Orange
27	9 (24)	INPUT 7	Red
28	9 (24)	LED 7	Brown
29	10 (25)	INPUT 8	Black
30	10 (25)	LED 8	White
37	8 (23)	INPUT 13	Gray
38	8 (23)	LED 13	Purple
39	7 (22)	INPUT 14	Blue
40	7 (22)	LED 14	Green
41	5 (20)	INPUT 15	Yellow
42	5 (20)	LED 15	Orange
43	6 (21)	INPUT 16	Red
44	6 (21)	LED 16	Brown

Table 2 *cont'd*: Pin connections for time delay modules

A Schematics for trigger circuitry.

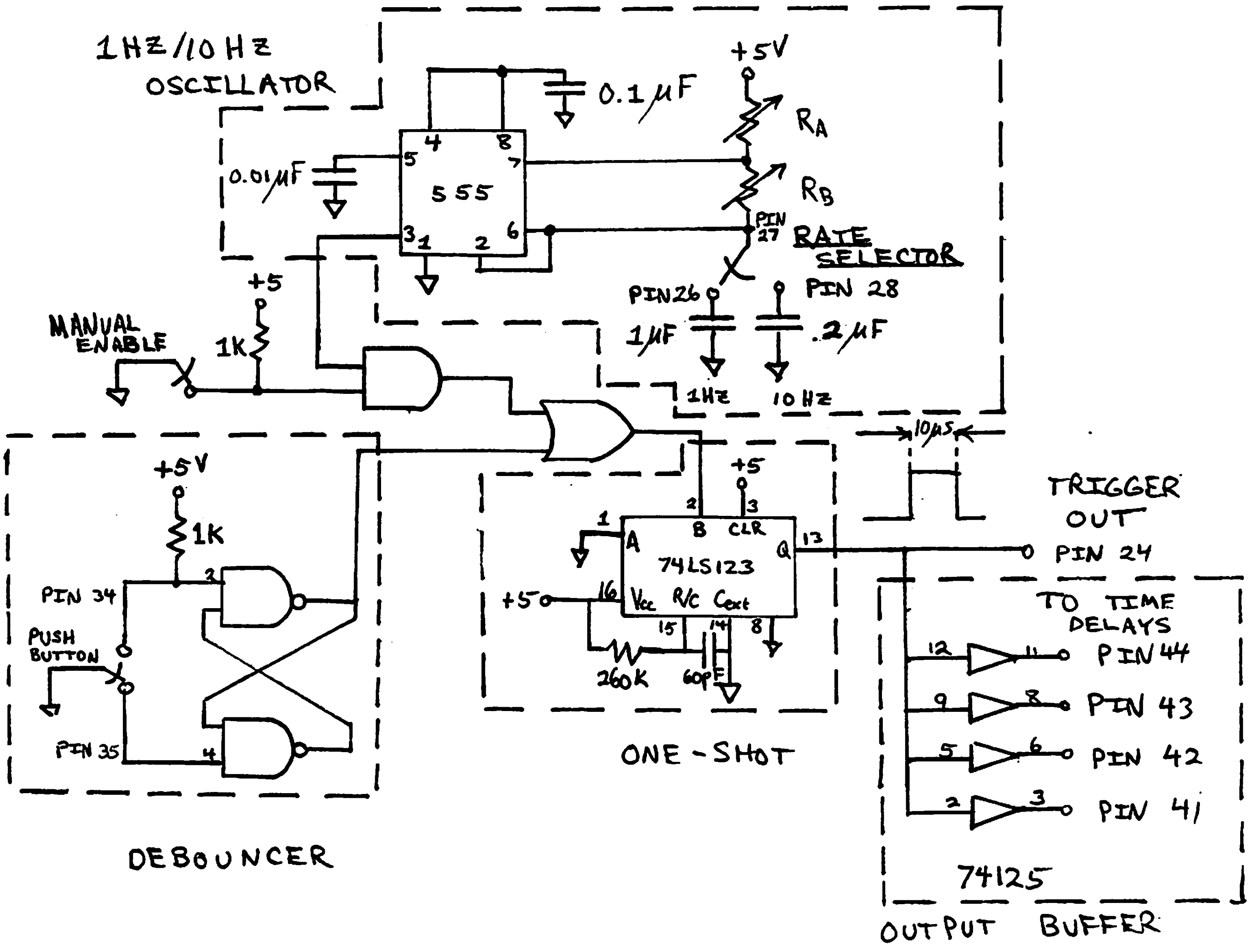


Figure 4: Trigger generator circuit.

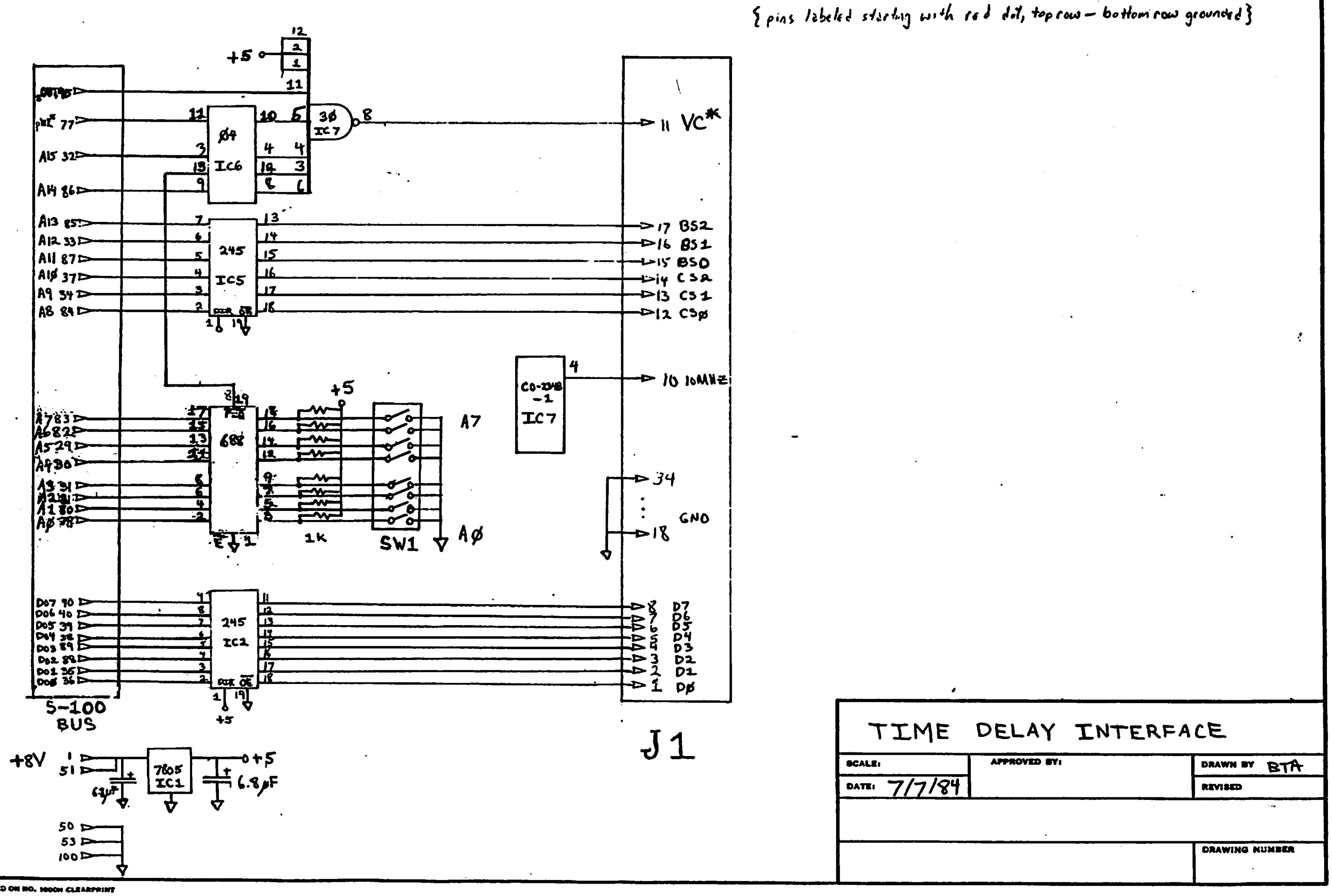
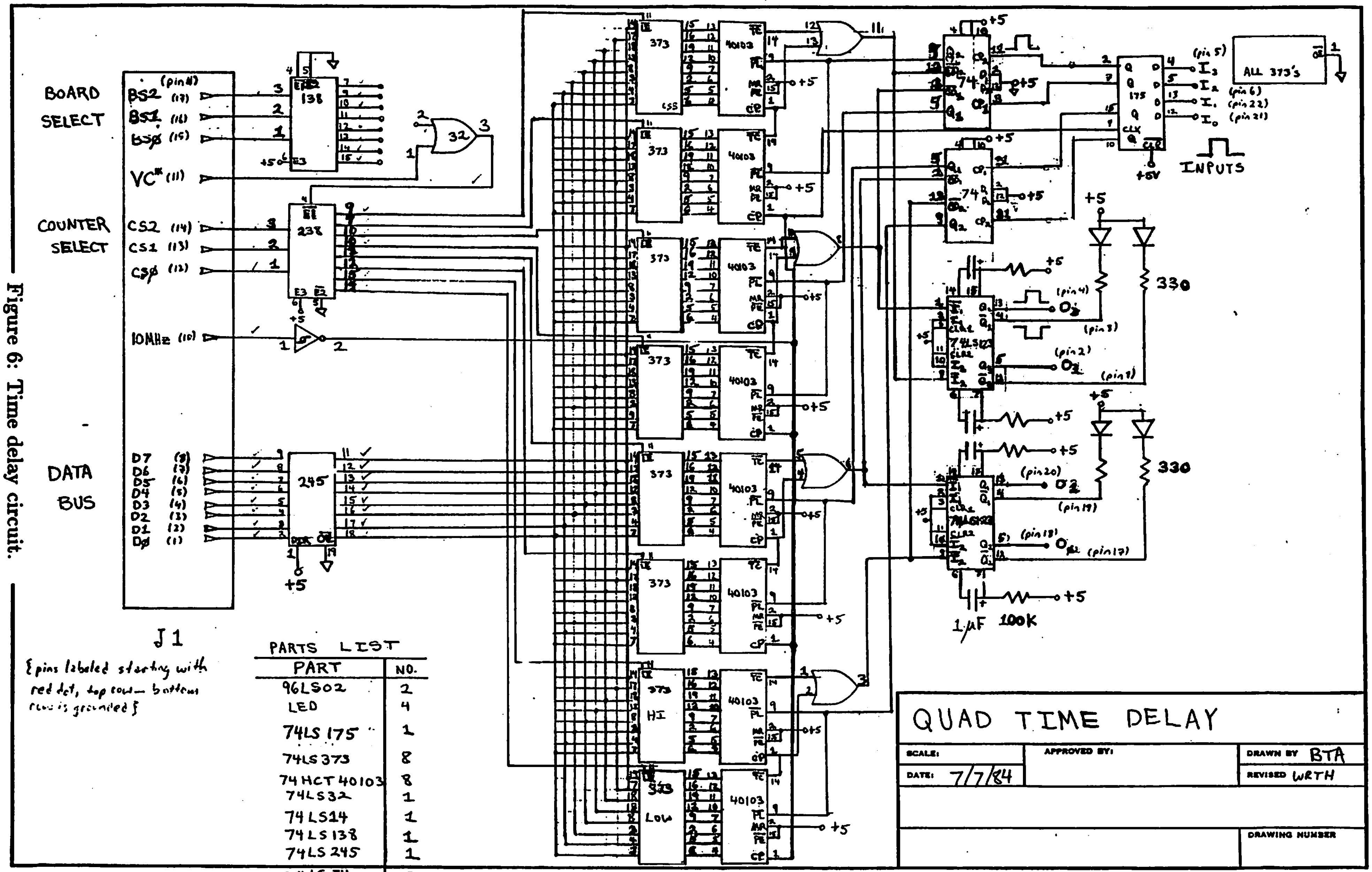


Figure 5: Time delay interface board.



PARTS LIST

PART	NO.
96LS02	2
LED	4
74LS175	1
74LS373	8
74HCT40103	8
74LS32	1
74LS14	1
74LS138	1
74LS245	1
74LS74	2
74LS238	1

QUAD TIME DELAY

SCALE:
APPROVED BY:
DRAWN BY BTA
DATE: 7/7/84
REVISED WRTH
DRAWING NUMBER

Figure 6: Time delay circuit.

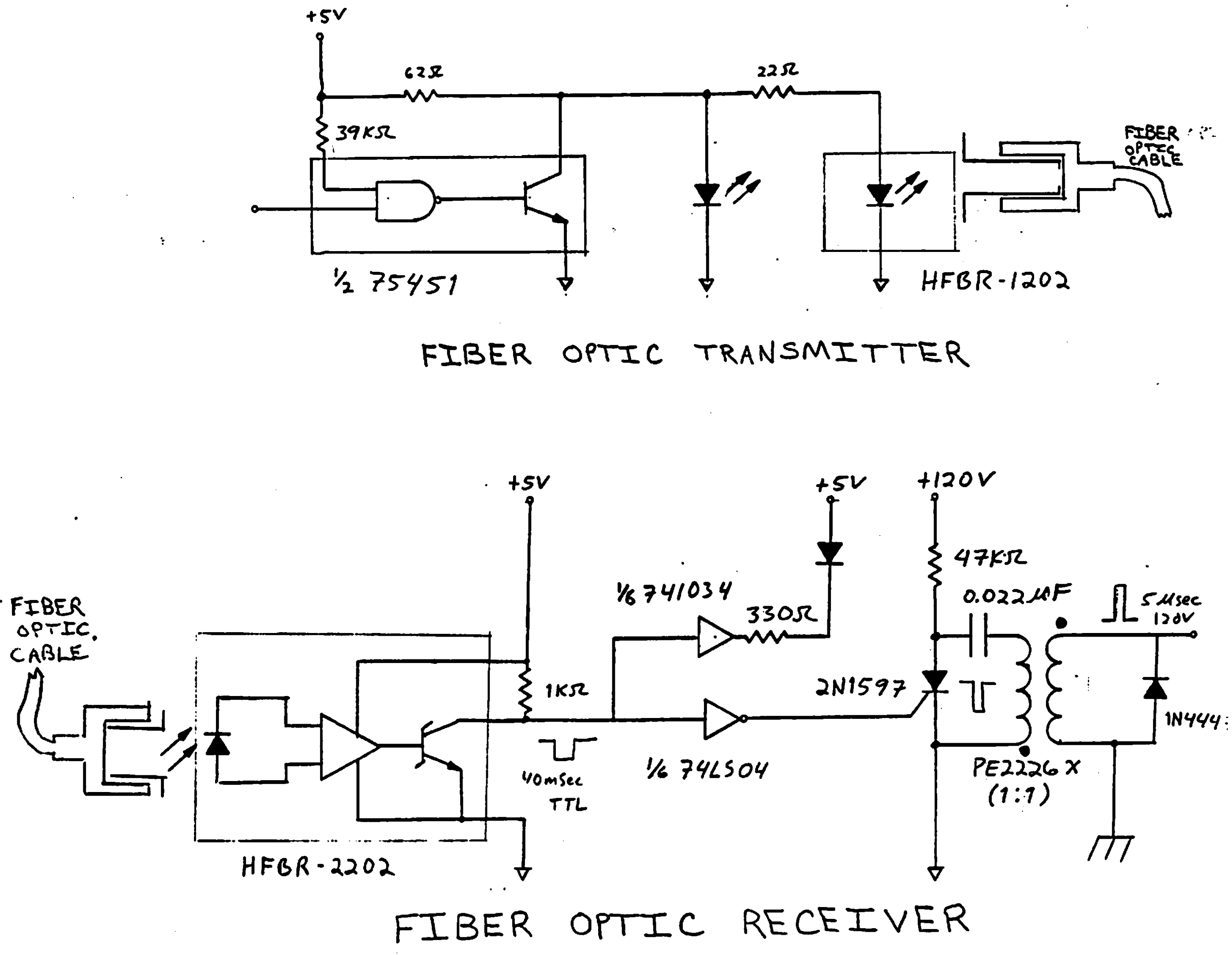


Figure 7: Optical coupler circuit.

B Listing of TD.C, the time delay controller code.

```
/*      TD.C -- TIME DELAY CONTROLLER
                   Branch T. Archer III
            Written: 1/31/85    Revised: 10/2/85
Compilation:
      c88 td
      bind td mmxlib.s              */

/*                INCLUDES                    */
#include "stdio.h"

#define MAXTD  32 /* MAXIMUM NUMBER OF TIME DELAYS */
#define TDNL    8      /* LENGTH OF TIME DELAY NAMES */
#define FIRSTROW 4     /* FIRST ROW OF DISPLAY */
#define LCOL 1         /* LEFT COLUMN OF DISPLAY */
#define RCOL 41        /* RIGHT COLUMN OF DISPLAY */
#define TDPORT 0xd0    /* BASE PORT FOR TIME DELAYS */
#define CLKFRQ 10.     /* CLOCK FREQUENCY */
#define FILE←LEN MAXTD*(TDNL+6+2) /* FILE LENGTH */

char file←name[MAXFNM],string[80],datim[6],rangerr[]="\7range error",
      data←format[]="%2d %-10.10s %8.1f %2d\r\n",tdname[MAXTD][TDNL+1],
      file←buf[FILE←LEN+1],chain←buf[30];
int fd,pfd,i,row,col,tdind,idt[6],tdtrig[MAXTD],buf←ptr;
unsigned clcndpts();
long filptr=512;
float tdset[MAXTD],oldset,deltat;
          /* DIGITIZER DATA FILE HEADER STRUCTURE */
struct HDR {char datim[6],filnam[11],name[32][7],slot[32],smpcod[32],
      reclen[32],pretrg[32],devtyp[32],crate[32],td[32],reserv[47]; } hdr;

main(argc,argv)
int argc;
char *argv[];
{cls(); strcpy(chain←buf,"DIGIT"); strcpy(file←name,"TIMING.DOC");
       /* GET PARAMETERS FROM COMMAND LINE */
switch (argc)
{      case 2: strcpy(file←name,argv[1]); break;
       case 3: sprintf(chain←buf,"DIGIT %s CED",argv[2]+5);
} /* END SWITCH */
if (fd=fopen(file←name,"r"))
      {if (argc==2) p←datfil(); /* Get data from digitizer data file */
      if (fread(file←buf,FILE←LEN,1,fd)!=1)
          {printf("Read error 1\n"); exit(0);} /* END IF */
          /* Load tdname array */
      for (tdind=buf←ptr=0; tdind<MAXTD; tdind++,buf←ptr+=TDNL)
          strncpy(tdname[tdind],&file←buf[buf←ptr],TDNL);
          /* Load tdset array, and set up time delays */
      for (tdind=0; tdind<MAXTD; tdind++,buf←ptr+=6)
          {sscanf(&file←buf[buf←ptr],"%6d",&i);
              tdset[tdind]=((float)i)/CLKFRQ;
              settd(tdind,tdset[tdind]); } /* END IF */
          /* Load tdtrig array */
      for (tdind=0; tdind<MAXTD; tdind++,buf←ptr+=2)
          sscanf(&file←buf[buf←ptr],"%2d",&tdtrig[tdind]);
```

```
/*    sscanf(&file←buf[buf←ptr],"%2d%2d%2d%2d%2d%2d",
          &idt[0],&idt[1],&idt[2],&idt[3],&idt[4],&idt[5]);
      for (i=0; i<6; i++) datim[i]=idt[i];    */
      fclose(fd);
 } /* END IF */

for (;;) {
scrmove(1,20); printf("*** TIME DELAY CONTROLLER ***");
scrmove(2,10); printf("FILE NAME: %s",file←name);
scrmove(3,LCOL); printf("TD NAME       SETTING (usec) TRIG");
scrmove(3,RCOL); printf("TD NAME       SETTING (usec) TRIG");
row=FIRSTROW; col=LCOL;
for (i=0; i<MAXTD; i++)
      {scrmove(row++,col);
      printf("%2d %-8.8s   %8.1f        %2d",i+1,tdname[i],tdset[i],tdtrig[i]);
      if (i==15) {row=FIRSTROW; col=RCOL;} /* END IF */
      } /* END FOR */
while (getind()) modtd();
scrmove(23,10);
printf("File name to write (<CR> returns to DIGIT, * to print): ");
string[0]=0;
scanf("%s",&string);
if (!string[0])
      {writetd(); printf(" CHAINING TO DIGIT . . ."); prgchn(chain←buf);}
if (!strcmp(string,"*")) prnttd();
      else {if (strlen(string)) strcpy(file←name,string); writetd();}
} /* END FOR */
return;
} /* END MAIN */

/*    POSITION FILE POINTER OF DIGITIZER DATA FILE TO TIME DELAY SETTINGS */
p←datfil()
{if (!fread(hdr,512,1,fd)) {printf("Read error 2\n"); exit(0);}
for (i=0; i<=32; i++)
      filptr+=clcndpts(hdr.slot[i],hdr.devtyp[i],hdr.reclen[i]);
fseek(fd,filptr,0);
} /* END P←DATFIL */

/*          WRITE TIME DELAY SETTINGS TO DISK  */
writetd()
{if (strlen(file←name))
{scrmove(23,10); printf("\33TWriting file: %s",file←name);
←setmem(file←buf,MAXTD*TDNL,' ');
      /* Copy time delay names into file buffer */
for (i=buf←ptr=0; i<MAXTD; i++,buf←ptr+=TDNL)
            strcpy(&file←buf[buf←ptr],tdname[i]);
      /* Replace nulls in the buffer with spaces */
for (buf←ptr=0; buf←ptr<(MAXTD*TDNL); buf←ptr++)
            if (file←buf[buf←ptr]=='\0') file←buf[buf←ptr]=' ';
      /* Put the time delay settings into the buffer */
for (i=0; i<MAXTD; i++,buf←ptr+=6)
            sprintf(&file←buf[buf←ptr],"%06d",(int)(CLKFRQ*tdset[i]));
      /* Put the triggers into the buffer */
for (i=0; i<MAXTD; i++,buf←ptr+=2)
```

```
                sprintf(&file←buf[buf←ptr],"%02d",tdtrig[i]);
        /* Put the date into the buffer */
/*datime(datim);
sprintf(&file←buf[buf←ptr],"%2d%2d%2d%2d%2d%2d",
        datim[0],datim[1],datim[2],datim[3],datim[4],datim[5]);
buf←ptr+=12;*/
        /* Open the file, write the data, and close */
fd=fopen(file←name,"w"); fwrite(file←buf,buf←ptr,1,fd); fclose(fd);
} /* END IF */
} /* END WRITETD */

/*    GET INDEX FOR TIME DELAY TO BE MODIFIED  */
getind()
{
scrmove(21,15); printf("\33TTime delay to modify: ");
tdind=0; getsrc(string,21,37,2);
if (strlen(string)) sscanf(string,"%d",&tdind);
printf("%2d",tdind);
if (tdind<0 || tdind>MAXTD)
        {scrmove (22,20); printf(" Invalid value.");
            tdind=-1; return(-1);} /* END IF */
scrmove (22,20); printf("\33T"); return(tdind);
} /* END GETIND */

/*    MODIFY TIME DELAY */
modtd()
{if (tdind<=0) return;
row=tdind+FIRSTROW-1;
if (tdind>16) {row-=16; col=RCOL;} else col=LCOL;
oldset=tdset[--tdind];
            /* Get time delay name. */
getsrc(string,row,col+3,TDNL);
if (strlen(string)) strcpy(tdname[tdind],string);
printf("%-8.8s",tdname[tdind]);
            /* Get time delay value. */
do     {
        getsrc(string,row,col+14,8);
        if (strlen(string)) sscanf(string,"%f",&tdset[tdind]);
        printf("%8.1f",tdset[tdind]);
        } /* END DO */
while (tdset[tdind]<0. || tdset[tdind]>65535./CLKFRQ);
            /* Get logical trigger for time delay. */
do     {
        getsrc(string,row,col+29,2);
        if (strlen(string)) sscanf(string,"%d",&tdtrig[tdind]);
        printf("%2d",tdtrig[tdind]);
        } /* END DO */
while ( tdtrig[tdind]<0 || tdtrig[tdind]>MAXTD || tdtrig[tdind]==tdind+1 );
settd(tdind,tdset[tdind]);
/* Modify time delays which are logically triggered by delay 'tdind'. */
/*scrmove(22,1); printf("\7Modifying time delays: ");*/
deltat=tdset[tdind]-oldset; modlc(tdind);
} /* END MODTD */
```

```
/* MODIFY LOGICAL CHAIN OF TIME DELAYS CONNECTED TO tdind */
/* This is a recursive routine which adds deltat to all time delays
connected to time delay 'tdind'. */
modlc(tdind)
int tdind;
{int i;
for (i=0; i<MAXTD; i++)
      {if (tdtrig[i]==tdind+1)
          {/*printf("%d",i+1);*/ tdset[i]+=deltat;
                  /* Check for range error. */
          if (tdset[i]<0.) {tdset[i]=0.; printf (rangerr);}
          if (tdset[i]>65535./CLKFRQ) {tdset[i]=65535./CLKFRQ; printf(rangerr);}
          row=FIRSTROW+i; if (i<16) col=LCOL; else {row-=16; col=RCOL;}
          scrmove(row,col+14); printf("%8.1f*",tdset[i]);
          settd(i,tdset[i]); modlc(i);
          } /* END IF */
      } /* END FOR */
} /* END MODLC */

/*    SET TIME DELAY */
/* 0 <= tdnum <= 31 */
settd(tdnum,tdval)
int tdnum;
float tdval;
{unsigned tdint;
tdint=CLKFRQ*tdval;/* printf("\n%x",tdint);*/
tdnum*=2;
/*    SEND THE HI BYTE    */
←outb(tdint>>8,TDPORT+(tdnum<<8));
/*    SEND THE LO BYTE    */
←outb(tdint,TDPORT+((tdnum+1)<<8));
} /* END SETTD */

/*    PRINT TIME DELAY SETTINGS    */
prnttd()
{pfd=fopen("PRN:","w");
sprintf(string,"MMX TIME DELAY SETTINGS %d/%d/%d %d:%d:%d\n",
      datim[0],datim[1],datim[2],datim[3],datim[4],datim[5]);
prntline();
strcpy(string,"  TD NAME      SETTING      TD NAME      SETTING\n");
prntline();
for (tdind=0; tdind<16; tdind++)
      {sprintf(string,"  %2d %-8.8s   %8.1f   %2d %-8.8s   %8.1f",
          tdind+1, tdname[tdind],tdset[tdind],
          tdind+16,tdname[tdind+16],tdset[tdind+16]);
      prntline();
      } /* END FOR */
fclose(pfd);
} /* END PRNTTD */

/*    PRINT A LINE, ALONG WITH <CR><LF>       */
prntline()
{strcat(string,"\n"); fputs(string,pfd);
} /* END PRNTLINE */
```

C Listing of TDCHECK.C, the time delay checking code.

```
/*          TDCHECK.C               BTA III
       WRITTEN: 2/16/85         REVISED: 2/19/85
Compilation:
       c88 tdcheck
       bind tdcheck camac mmxlib

Command line:
       tdcheck <haltflag> <file name>
If <haltflag> == HALT, the program aborts after an error.
<file name> is the name of the error file (e.g. PRN: or CON:).

       This code checks a given time delay channel for the proper
delay.  The number of codes to skip between checks is adjustable.
The time delay is checked by using a signal generator to trigger
a time delay and a digitizer.  The output of the time delay is fed
into the digitizer as the signal.  This code then measures the delay
by scanning for the first point in the digitizer data array which
is saturated.
*/

#include "stdio.h"
#define TDPORT 0xd0   /* BASE PORT FOR TIME DELAYS */
#define CLKFRQ 10.
#define MAXERRS 100

char data[8192],string[80],datim[6],file←name[MAXFNM]="CON:";
int pfd,tdnum,slot,crate=0,smpint,reclen,pretrg,devtyp,skip,satpos,
       rec8192[3]={0,2,7},abortflag;
unsigned hexdelay,firstdelay,checknum,numchecks,numerrs=0;
float fldelay,msdelay,maxst,maxerr;
float smptim[3][8]=
            {.05,.10,.20,.50,1.0,2.0,5.0,0.0,  /* TRANSIAC 2008  */
             .01,.02,.05,.10,.20,.50,1.0,0.0,  /* TRANSIAC 2001  */
             .03125,.0625,.125,.25,.5,1.,2.,0.};/* LeCroy   8837F */
float smplen[3][7]=
            {350.,700.,1400.,4000.,8000.,16000.,40000.,
             70.,140.,350.,700.,1400.,4000.,8000.,
             200.,400.,1000.,2000.,4000.,8000.,16000.};

main(argc,argv)
int argc;
char *argv[];
{if (argc>1) abortflag=!strcmp("HALT",argv[1]);
if (argc>2) strcpy(file←name,argv[2]);
cls(); scrmove(1,20); printf("*** TIME DELAY CHECK ***");
/*          GET TIME DELAY TO BE CHECKED         */
scrmove(5,10); printf("Time delay to check:");
for (;;)
       {scrmove(5,31); screteol(); scanf("%d",&tdnum);
            if (tdnum<1 || tdnum>32)
                {scrmove(7,10); printf("Invalid delay number.\7");}
            else break;} /* END FOR */

/*          GET DIGITIZER TO BE USED              */
```

```
scrmove(7,10); screteol(); printf ("Digitizer slot to be used:");
for (;;)
        {scrmove(7,37); scanf("%d %d",&slot,&crate);
        digrd(crate,slot,&smpint,&reclen,&pretrg,&devtyp);
            if (!devtyp)
                {scrmove(9,10); printf("Digitizer unavailable.\7");}
            else break;} /* END FOR */
/*    SET RECLEN TO THE VALUE WHICH GIVES 8192 POINTS */
reclen=rec8192[devtyp-1];
/*    SELECT 0 PRE-TRIGGER SAMPLES            */
pretrg=0;
/*    GET FIRST DELAY VALUE, AND NUMBER TO BE SKIPPED */
scrmove(9,10); screteol();
printf("First setting (1-65535), and skip value:");
for (;;)
        {scrmove(9,51); scanf("%d %d",&firstdelay,&skip);
        if (skip<0)
                {scrmove(11,10); printf("Bad skip value.\7");}
        else break;} /* END FOR */
scrmove(11,10);
numchecks = skip ?
(float)((unsigned)0xffff-firstdelay)/(float)(skip) : 0xffff;
printf("Number of checks to be performed: %u\n",numchecks);
/*    INITIALIZE DELAY VALUE, AND MAXIMUM SAMPLING TIME */
hexdelay=firstdelay;
maxst=0.;
pfd=fopen(file←name,"w");
sprintf(string,"   Error log for time delay #%d\n",tdnum);
prntline();
sprintf(string,"   Slot %d    Devtyp = %d\n",slot,devtyp);
prntline();
datime(datim);
sprintf(string,"   Date: %d/%d/%d    Time: %d:%d:%d\n",
        datim[0],datim[1],datim[2],datim[3],datim[4],datim[5]);
prntline();
sprintf(string,"   First value = %u   Skip count = %u",firstdelay,skip);
prntline();
for (checknum=1; checknum<=numchecks; checknum++)
        {checktd(); hexdelay+=skip;} /* END FOR */
datime(datim);
sprintf(string,"All done at %d:%d:%d",datim[3],datim[4],datim[5]);
prntline();
sprintf(string,"--- %u errors ---",numerrs); prntline();
fclose(pfd);
} /* END MAIN */

/*        CHECK A TIME DELAY                */
checktd()
{if ((fldelay=(float)hexdelay/CLKFRQ)>maxst) setsmp();
settd(tdnum-1,fldelay);
digint(crate,slot);
digdat(crate,slot,&data,0);
satpos=scasb(data,0xff,8192);
if (satpos==8192)
```

```
/*        PRINT ERROR MESSAGE     */
{sprintf(string,"     No saturation for setting %8.1f",fldelay);
prntline(); numerrs++; if (numerrs==MAXERRS) abortprg();
if (abortflag) abortprg(); return;}
msdelay=smptim[devtyp-1][smpint]*(float)satpos;
printf("Delay setting = %8.1f  Measured setting = %8.1f\n",
      fldelay,msdelay);
if (ABS(fldelay-msdelay)>maxerr)
/*        PRINT ERROR MESSAGE     */
{sprintf(string,
"     Error for setting %8.1f(uS) %4x. Measured value = %8.1f(uS).",
      fldelay,hexdelay,msdelay); prntline(); ++numerrs;
      if (numerrs==MAXERRS) abortprg();
      if (abortflag) abortprg();}
} /* END CHECKTD */

/*        SET SAMPLING RATE FOR DIGITIZER    */
setsmp()
{if (fldelay>smplen[devtyp-1][6])
      {printf("Delay value of %8.1f too large.\n"); exit();}
/*    SELECT FASTEST POSSIBLE SAMPLING RATE  */
for (smpint=0; smpint<=7; smpint++)
      {if (smplen[devtyp-1][smpint]>fldelay) break;}
maxst=smplen[devtyp-1][smpint];
maxerr=MAX(3.*smptim[devtyp-1][smpint],.25);
sprintf(string,
"Setting sampling rate. Smpint = %d Maxerr=%f",smpint,maxerr);
prntline(); printf("%s",string);
digset(crate,slot,smpint,reclen,pretrg,devtyp);
} /* END SETSMP */

/*    SET TIME DELAY */
/* 0 <= tdnum <= 31 */
settd(tdnum,tdval)
int tdnum;
float tdval;
{unsigned tdint;
tdint=CLKFRQ*tdval;
tdnum*=2;
/*    SEND THE HI BYTE    */
{outb(tdint>>8,TDPORT+(tdnum<<8));
/*    SEND THE LO BYTE    */
{outb(tdint,TDPORT+((tdnum+1)<<8));
} /* END SETTD */

/*        ABORT PROGRAM        */
abortprg()
{sprintf(string,"Aborting with %d errors.",numerrs);
prntline(); exit();
} /* END ABORTPRG */

/*            MOVE CURSOR     */
scrmove(row,col)
int row,col;
```

```
{printf("\33=%c%c",row+' '-1,col+' '-1);
 } /* END SCRMOVE */

/*         ERASE TO END OF CURRENT LINE        */
screteol()
{printf("\33T");
 } /* END SCRETEOL */

/*              CLEAR SCREEN          */
cls()
{printf("\33*");
 } /* END SCRCLEAR */

/*    PRINT A LINE, ALONG WITH <CR><LF>        */
prntline()
{strcat(string,"\n");
fputs(string,pfd);
 } /* END PRNTLINE */

```