MEMO-3-83

EIGEN LINK to VAX

J. L. Turney

September 1983

CENTER FOR ROBOTICS AND INTEGRATED MANUFACTURING

Robot System Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109

The Eigen Link to the VAX

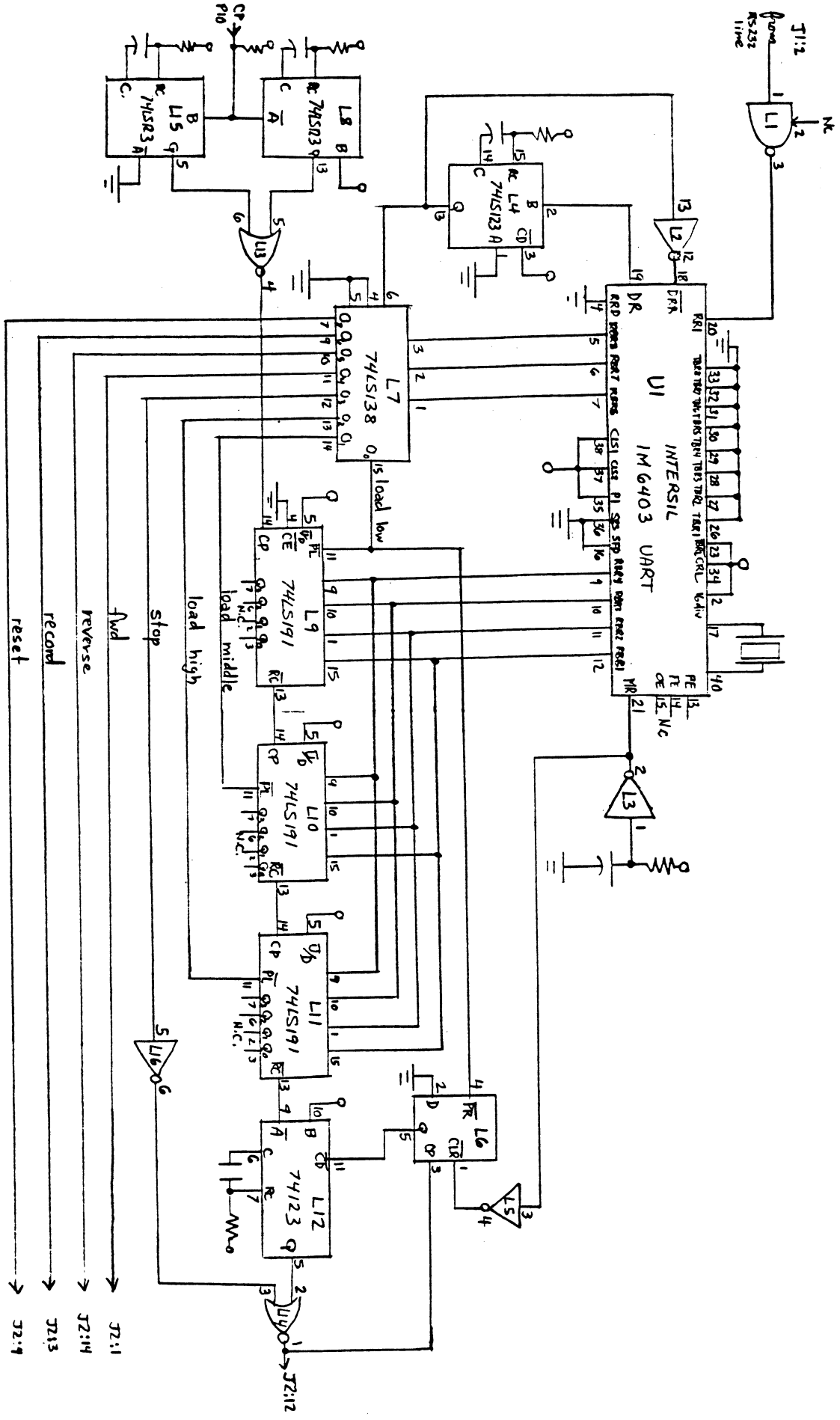
The eigen link is a software/hardware interface which allows the user to control the Eigen Video video disc recorder from the VAX. The eigen link consists of a simple software driver on the VAX, a 9600 baud serial line, and a small gray box located near the Eigen Video controller which contains the hardware interface from the VAX to the Eigen Video controller.

The eigen link functions as follows. The software driver accepts single charascter commands with disc track numbers or disc track offsets from the user. It translates these commands to a 9600 baud serial command signal to the hardware interface. The hardware receives the 9600 baud serial command signal, decodes the command, and loads offsets into a set of counters.

Hardware Description

Fig. 1 displays a schematic for the hardware in the eigen link. The heart of design is the Intersil IM6403 UART, "Universal Asynchronous Receiver and Transmitter". L3 initializes the UART on power up. L1 converts the input serial signal from the VAX into a TTL level input into the UART. The signal is received serially at 9600 baud, 7 bits per frame, with the parity bit ignored. The UART has its own internal baud rate clock. When a frame of data has been received, DR at pin 19 of the UART goes high triggering the one shot L4 and reseting the receive buffer through L2. L7 decodes the upper three bits of the frame. The decoded signal from L7 either loads one of the three 4-bit counters, L9, L10, or L11, or to sends a command to the eigen controller. The counting mechanism is armed by setting L6. If L6 is not set, the eigen controller will run until stopped by the stop command. The counters L9-L11 are set to count down, so

that once they are loaded, they will accept clock pulses from the one shots L8 and L15 and count down to zero. Once they hit zero they send a stop command through L14 if L6 is armed. The two one shots L8 and L15 are used to trigger off the leading and trailing edge of the clock output from the eigen controller.


J1:2
from RS232 line
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
L13
L15
U1
IM6403 UART
INTERSIL
DR
load low
load high
load middle
stop
fwd
reverse
record
reset
J2:1
J2:14
J2:13
J2:9
J2:12

Software Description A description of the software is presented in the standard manual form for UNIX on the next page.

NAME

eigen – invoke the eigen software interface

SYNOPSIS

eigen

DESCRIPTION

eigen invokes the software interface to the Eigen Video controller. The software interface accepts single character commands followed by either absolute track locations on the video disc or by track offsets.

The syntax for a command is simply:

<*command character*>[<*offset*> | @<*track*>]

where the square brackets "[ ]" indicate that the enclosed items are optional, and the "|" separates the options. <*command character*> is one of the command characters, {+|-|r|s|t|h|q}, If the "@" character is included, <*track*> which must follow refers to an absolute track location, if the "@" character is absent then <*offset*> refers to a relative track location. No spaces are allowed between characters. Below are displayed the commands and their actions.

+	forward
+<*offset*>	forward for <*offset*> tracks
+@<*track*>	forward until <*track*> location
-	backward
-<*offset*>	backward for <*offset*> tracks
-@<*track*>	backward until <*track*> location
r	record
r<*offset*>	record for <*offset*> tracks
r@<*track*>	record until <*track*> location
s	stop
t	display current track
t<*track*>	set internal track count to <*track*>
h	help by displaying command summary
q	quit

BUGS

When the Eigen Video video disc is powered up, it does not initialize to the same starting point so that an absolute track position is not *really* absolute, but only absolute with respect to the current session.

Appendix

```
#include <stdio.h>

#define WRITE_MODE 1

#define END_OF_STRING 0

#define TRUE 1

#define FALSE 0

#define SUCCESS  1

#define FAILURE 0

#define PROCEDURE

#define DO_INDEFINITELY while(1)


char load_low        = 0x00,
     load_middle     = 0x20,
     load_high       = 0x40,
     stop            = 0x60,
     forward         = 0x80,
     reverse         = 0xA0,
     record          = 0xC0,
     reset           = 0xE0;

int eigen_line, i;


PROCEDURE main() {
   int number_track, attempt, track, count,
      nargs, sync, temp, absolute;
   char command_line[20], input_char;
```

```
/*****************************************
tty18 is the line dedicated to the eigen link
*****************************************/

eigen_line = open( "/dev/tty18", WRITE_MODE );


/***************************
Zero the counter of the eigen
***************************/

 write( eigen_line, &reset, 1 );

track = 0;

sync = TRUE;     /*** eigen is synchronized ***/




DO_INDEFINITELY
{

   next_command: printf("0igen: ");

   i=0;

   while( (input_char = getchar()) != '0 && i<19 )

      command_line[i++] = input_char;

   command_line[i] = 0;


   number_track = 0;



   count = FALSE;

   absolute = FALSE;  /*** input track numbers can be for
                          absolute addresses or relative
                          addresses ***/




   /*****************
   Input eigen command
   *****************/
```

```
if( (command_line[1] != ' ' ) &&
   (command_line[1] != ' '  ) )
{

   nargs = sscanf( command_line, "%*1c%d", &number_track );

   if( nargs<=0 )  /*** If command line did not fit above
                       format try a different format with
                       an absolute address for the track
                       number specified ***/
   {
      nargs =sscanf( command_line,"%*1c@%d",&number_track );

      /*** look only for the track number ***/

      if( nargs==1 )
      {
         if( sync == FALSE ) /*** if an absolute track
                                  number is specified but
                                  the eigen link has lost
                                  count of where it is, then
                                  report that a new absolute
                                  address must be loaded ***/

         {
            printf("0rror: track count lost, must reinit0);
            goto next_command;
         }

         absolute = TRUE;    /*** an absolute address has
                                  been specified ***/
      }
   }

   if( command_line[0] == '+' ||
      command_line[0] == '-' ||
      command_line[0] == 'r' )
   {

      if( absolute ) /*** if an absolute address was
                         specified then find the difference
                         between the current location and
                         the desired absolute location and
                         load this to the 12 bit down counter
                         on the eigen link ***/
      {
         temp = number_track;

         switch( command_line[0] )
         {
            case '+':

            case 'r':

               number_track = ( number_track - track
```

```
                        + 600 )%600;
            break;

            case '-':

               number_track = (track - number_track

                        + 600)%600;
            break;

         }
      }


      attempt = load_count( number_track );  /*** load count
                                              here ***/

      if( attempt == FAILURE ) goto next_command;

      count = TRUE;

      switch( command_line[0] ) /*** adjust the track
                              address appropriately **/
      {

      case '+' :

      case 'r' : if( absolute ) track = temp;

         else track = ( track + number_track )%600;

      break;

      case '-' : if( absolute ) track = temp;

         else track = ( track - number_track + 600 )%600;

      break;

      }
   }

}

if( command_line[1] == ' ' ) command_line[0] = '?';



/*****************************************************
Command handler. Send eigen commands to the eigen link
*****************************************************/

switch( command_line[0] )
```

```
{

   /***********************************************
   If no count is specified for the "+", "-" or "r"
   command, or if the s command is executed, this
   program loses the correct track address and must
   be recalibrated with the "t" command.
   ***********************************************/

   case '+': write( eigen_line, &forward, 1 );

      if( count == FALSE ) sync = FALSE;


   break;


   case '-': write( eigen_line, &reverse, 1 );

      if( count == FALSE ) sync = FALSE;

   break;


   case 'r': write( eigen_line, &record, 1 );

      if( count == FALSE ) sync = FALSE;

   break;


   case 's': write( eigen_line, &stop, 1 );

      if( count == FALSE ) sync = FALSE;

   break;



   /*********************************************************
   Print current track number if the command has no number
   specified. Set the track address if a count is specified
   *********************************************************/

   case 't': if(command_line[1]==' ')
         {

            if( sync == FALSE )
            {
               printf("0rror: track count lost, must reinit0);
               goto next_command;
            }


            printf("0urrent track: %3d0, track );
```

```
            }
            else
            {
               track = number_track;

               printf("0rack reinitialized0);

               sync = TRUE;
            }

         break;


         case 'h':


            printf("   +              forward");

            printf("   +CNT               forward for CNT tracks");
            printf("   +@TRACK                forward to TRACK");

            printf("   -              reverse");

            printf("   -CNT               reverse for CNT tracks");
            printf("   -@TRACK                reverse to TRACK");

            printf("   r              record");

            printf("   rCNT               record for CNT tracks");
            printf("   r@TRACK                record to TRACK");

            printf("   s              stop eigen");

            printf("   t                 print current track number");
            printf("   tTRACK                set track number to TRACK");

            printf("   h              help");

            printf("   q              quit0);

         break;


         case 'q': exit(0);


         default: printf("0rror: unrecognizable command, type h for help0);

      }

   }

}
```

```
int PROCEDURE load_count( number_track )

int number_track; {

   char high_byte, middle_byte, low_byte, count;

   int i;


   if( (number_track < 0) || (number_track >= 4096) )
   {
      printf("Orror: argument out of range, 0-40950);

         return( FAILURE );

   }

   if( number_track == 0 ) return(FAILURE);




   /*****************************************************
   The number of tracks to be skipped is broken into
   three nibbles. These are combined with commands which
   allow three 4-bit counters to be loaded with the
   number of tracks to be skipped. The counters
   will be decremented to zero. When the count
   goes negative the eigen will be stopped by the
   counters. The number of tracks is therefore
   loaded with one less than the desired number of
   tracks.
   *****************************************************/

   number_track--;


   count = number_track/256;

   high_byte = load_high | count;

   number_track -= count*256;


   count = number_track/16;

   middle_byte = load_middle | count;

   number_track -= count*16;


   count = number_track;
```

```
    low_byte = load_low | count;


    /***********************************
    The three 4-bit counters are loaded.
    ***********************************/

    write( eigen_line, &low_byte, 1 );

    i=0;
    while(i++<1000);

    write( eigen_line, &middle_byte, 1 );

    i=0;
    while(i++<1000);

    write( eigen_line, &high_byte, 1 );

    i=0;
    while(i++<1000);

    return( SUCCESS );

}
```