AN4000/D

Visual Debug for MPC60x

John Ralston, Motorola Ltd., East Kilbride, Scotland

1.0 Introduction

When a new MPC60x system design is in the initial debug phase there is often no simple means of providing feedback to the human debugger. To overcome this, the circuit described below was implemented. It allows the display of two hex digits by writing or reading memory locations. No initialization/setup is required.

A simple debug scenario could be that the initial code that the 60x executes just displays a incrementing number on the hex display. This would show that the code source, the 60x and the hex display hardware are functional. This could then be expanded as functional block tests are designed. For example, table 1 gives an example display sequence for a memory test.

Table 1. An example display sequence for memory test

Hex number	Description
01	60x can talk to hex display
10	started memory test
11	write first pattern to memory
12	read first pattern from memory and check it
13	write second pattern to memory
14	read second pattern from memory and check it.
18	finished OK
91	error on read of first pattern
92	error on read of second pattern

2.0 Theory of Operation

The design only uses the address phase of the MPC60x bus and requires the target system to terminate the cycle. The reason for using only the address phase signals is that the MPC60x bus is able to pipeline two addresses and have external reordering of data phases to the addresses. This would require logic to determine the correct address/data relationship. The design uses the following signals out of the address tenure, **A(0:31)**, $\overline{\textbf{TS}}$, **TT(3:4)**, and the CPU clock (**CLK**). This means that either a load or a store can be used to communicate the code value. The board operates by decoding the top 24 address lines to give a unique position in the address space. It then latches the 8 least significant addresses of the bus and displays them in hex on two 7-segment displays. The decode position was chosen to be 0x000000xx, as it is one of the address ranges that should always be active after reset without program intervention and it is a region that would not normally be used. If 0x000000xx is not suitable, then 0xFFF000xx should be considered.

MOTOROLA

3.0 USAGE

The board accessed by any load or store in the range 0x00000000 - 0x000000FF. The following example code will transfer codes to be displayed on the board.

C code example:

```
#define US8 unsigned char            /* 8 bit unsigned value                  */

display(US8 code_value)
{
   US8 *address;                     /* pointer to byte location              */
   address = (US8 *) code_value;     /* assumes zero fill by cast operation   */
   *address = 0x00;                  /* store a dummy value                   */
}
```

Assemble code example:

```
   xor      r4,r4,r4                 # zero r4
   ori      r4,r4,code_value         # code_value is 0x00 || 8 bit code
   stb      r4,0x00(r4)              # store dummy value
```

4.0 Circuit Description

The circuit consists of three 22v10 PLDs and two 7-segment displays as shown in figure 1. The PLD equations are at the end of this document. Two of the PLDs, SEGMSB (U2) and SEGDEC (U3), are used to perform the hex (4-bit) to 7-segment decode (a, b, c, d, e, f and g). They update the display each time **load_pulse** goes through a low to high transition. In addition SEGDEC also performs a predecode on **TT(3:4)** and **A(0:6)** to generate **$\overline{\text{HI\_DECODE}}$** asynchronously. The last PLD, DEC (U1), takes in **$\overline{\text{TS}}$, $\overline{\text{HI\_DECODE}}$, A(7:23)**, and the CPU clock (CLK) to generate **LOAD_PULSE** for SEGMSB and SEGDEC.

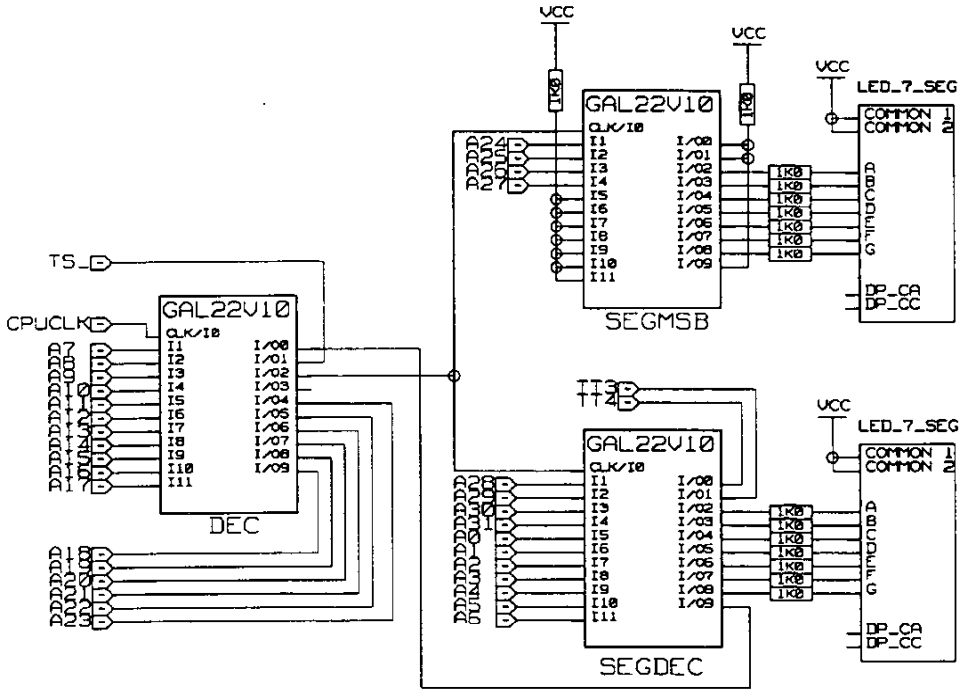

Figure 1. Circuit Diagram

5.0 PLD Equations

5.1 DEC.PLD

```
Name        dec;
Partno      P00001;
Date        29/07/94;
Revision    01;
Designer    jwr (EKB);
Company     Motorola Ltd Copyright (C);
Assembly    Display Board;
Location    U1;
Device      p22v10;
Format      j;

/*************************************************************************/
/*                                                                       */
/* DESCRIPTION:                                                          */
/* Master decode for Hex board                                           */
/*                                                                       */
/* HISTORY:                                                              */
/*                                                                       */
/* V01 290794 JWR Initial Version                                        */
/*                                                                       */
/*************************************************************************/
/* Allowable Target Device Types: 22V10 10ns or faster                   */
/*************************************************************************/

/** Inputs **/

Pin 1          = Sys_cp;          /* System clock          */
Pin [2..11]    = [s_a7..16] ;     /* System address        */
Pin [13..19]   = [s_a17..23] ;    /* System address        */
Pin 22         = !ts ;            /* Transfer start        */
Pin 23         = !hi_decode ;     /* Top decode            */

/** Outputs **/

Pin 20         = !delayed_ts ;    /* Delayed to allow for hi_decode   */
Pin 21         = load_pulse  ;    /* Load 7 segment number            */

/** Declarations and Intermediate Variable Definitions **/

Field sys_addr = [s_a7..23] ;

/** Logic Equations **/

delayed_ts.d = ts ;

load_pulse.d = delayed_ts & hi_decode & sys_addr:'h'00000000 ;
```

```
/*** Enables ***/

load_pulse.oe = 'b'1;
delayed_ts.oe = 'b'1;

/** Resets **/

load_pulse.ar = 'b'0;
delayed_ts.ar = 'b'0;

/** Presets **/

load_pulse.sp = 'b'0;
delayed_ts.sp = 'b'0;
```

5.2 SEGDEC.PLD

```
Name        segdec;
Partno      P00001;
Date        29/07/94;
Revision    01;
Designer    jwr (EKB);
Company     Motorola Ltd Copyright (C);
Assembly    Display Board;
Location    U3;
Device      p22v10;
Format      j;

/*************************************************************************/
/*                                                                       */
/* DESCRIPTION:                                                          */
/*             Decode of top 7 address lines.                            */
/*             4 to 7 segment decoder.                                   */
/* HISTORY:                                                              */
/*                                                                       */
/* V01 290794 JWR Initial Version                                        */
/*                                                                       */
/*************************************************************************/
/* Allowable Target Device Types: 22V10 10ns or faster                   */
/*************************************************************************/
```

```
/** Inputs **/

Pin 1          = load_pulse ;     /* Load pulse              */
Pin [2..5]     = [a3..0] ;        /* Nibble to be displayed  */
Pin [6..11]    = [s_a0..5] ;      /* System address          */
Pin 13         = s_a6 ;           /* System address          */

/** Outputs **/

Pin 14         = !hi_decode  ;    /* Top decode              */
Pin 15         = !seg_g ;         /* Segments                */
Pin 16         = !seg_f ;         /* Segments                */
Pin 17         = !seg_e ;         /* Segments                */
Pin 18         = !seg_d ;         /* Segments                */
Pin 19         = !seg_c ;         /* Segments                */
Pin 20         = !seg_b ;         /* Segments                */
Pin 21         = !seg_a ;         /* Segments                */
Pin 22         = TT3;             /* Transfer Type 3         */
Pin 23         = TT4 ;            /* Transfer Type 4         */

/** Declarations and Intermediate Variable Definitions **/

Field addr = [a3..a0];

/** Logic Equations **/

hi_decode = !s_a0 & !s_a1 & !s_a2 & !s_a3 & !s_a4 & !s_a5 & !s_a6 & TT3 & !TT4;

seg_a.d        = addr:'h'0
               # addr:'h'2
               # addr:'h'3
               # addr:'h'5
               # addr:'h'6
               # addr:'h'7
               # addr:'h'8
               # addr:'h'9
               # addr:'h'a
               # addr:'h'c
               # addr:'h'e
               # addr:'h'f;

seg_b.d        = addr:'h'0
               # addr:'h'1
               # addr:'h'2
               # addr:'h'3
               # addr:'h'4
               # addr:'h'7
               # addr:'h'8
               # addr:'h'9
               # addr:'h'a
```

```
                # addr:'h'd;

seg_c.d         = addr:'h'0
                # addr:'h'1
                # addr:'h'3
                # addr:'h'4
                # addr:'h'5
                # addr:'h'6
                # addr:'h'7
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'b
                # addr:'h'd;

seg_d.d         = addr:'h'0
                # addr:'h'2
                # addr:'h'3
                # addr:'h'5
                # addr:'h'6
                # addr:'h'8
                # addr:'h'b
                # addr:'h'c
                # addr:'h'd
                # addr:'h'e;

seg_e.d         = addr:'h'0
                # addr:'h'2
                # addr:'h'6
                # addr:'h'8
                # addr:'h'a
                # addr:'h'b
                # addr:'h'c
                # addr:'h'd
                # addr:'h'e
                # addr:'h'f;

seg_f.d         = addr:'h'0
                # addr:'h'4
                # addr:'h'5
                # addr:'h'6
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'b
                # addr:'h'c
                # addr:'h'e
                # addr:'h'f;

seg_g.d         = addr:'h'2
```

```
                    # addr:'h'3
                    # addr:'h'4
                    # addr:'h'5
                    # addr:'h'6
                    # addr:'h'8
                    # addr:'h'9
                    # addr:'h'a
                    # addr:'h'b
                    # addr:'h'd
                    # addr:'h'e
                    # addr:'h'f;

/** Enables **/

seg_a.oe = 'b'1;
seg_b.oe = 'b'1;
seg_c.oe = 'b'1;
seg_d.oe = 'b'1;
seg_e.oe = 'b'1;
seg_f.oe = 'b'1;
seg_g.oe = 'b'1;

/** Resets **/

seg_a.ar = 'b'0;
seg_b.ar = 'b'0;
seg_c.ar = 'b'0;
seg_d.ar = 'b'0;
seg_e.ar = 'b'0;
seg_f.ar = 'b'0;
seg_g.ar = 'b'0;

/** Presets **/

seg_a.sp = 'b'0;
seg_b.sp = 'b'0;
seg_c.sp = 'b'0;
seg_d.sp = 'b'0;
seg_e.sp = 'b'0;
seg_f.sp = 'b'0;
seg_g.sp = 'b'0;
```

5.3 SEGMSB.PLD

```
Name        segmsb;
Partno      P00001;
Date        29/07/94;
Revision    01;
Designer    jwr (EKB);
Company     Motorola Ltd Copyright (C);
Assembly    Display Board;
Location    U2;
Device      p22v10;
Format      j;

/*************************************************************************/
/*                                                                       */
/* DESCRIPTION:                                                          */
/*             4 to 7 segment decoder.                                   */
/*                                                                       */
/* HISTORY:                                                              */
/*                                                                       */
/* V01 290794 JWR Initial Version                                        */
/*                                                                       */
/*************************************************************************/
/* Allowable Target Device Types: 22V10 10ns or faster                   */
/*************************************************************************/

/** Inputs **/

Pin 1          = load_pulse ;    /* Load pulse              */
Pin [2..5]     = [a3..0] ;       /* Nibble to be displayed  */
Pin [6..11]    = [IN6..11] ;     /* Spare inputs            */
Pin 13         = IN13 ;          /* Spare inputs            */

/** Outputs **/

Pin 14         = IO14 ;          /* Spare I/O               */
Pin 15         = !seg_g ;        /* Segments                */
Pin 16         = !seg_f ;        /* Segments                */
Pin 17         = !seg_e ;        /* Segments                */
Pin 18         = !seg_d ;        /* Segments                */
Pin 19         = !seg_c ;        /* Segments                */
Pin 20         = !seg_b ;        /* Segments                */
Pin 21         = !seg_a ;        /* Segments                */
Pin 22         = IO22 ;          /* Spare I/O               */
Pin 23         = IO23 ;          /* Spare I/O               */

/** Declarations and Intermediate Variable Definitions **/

Field addr = [a3..a0];
```

```
/** Logic Equations **/

seg_a.d         = addr:'h'0
                # addr:'h'2
                # addr:'h'3
                # addr:'h'5
                # addr:'h'6
                # addr:'h'7
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'c
                # addr:'h'e
                # addr:'h'f;

seg_b.d         = addr:'h'0
                # addr:'h'1
                # addr:'h'2
                # addr:'h'3
                # addr:'h'4
                # addr:'h'7
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'd;

seg_c.d         = addr:'h'0
                # addr:'h'1
                # addr:'h'3
                # addr:'h'4
                # addr:'h'5
                # addr:'h'6
                # addr:'h'7
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'b
                # addr:'h'd;

seg_d.d         = addr:'h'0
                # addr:'h'2
                # addr:'h'3
                # addr:'h'5
                # addr:'h'6
                # addr:'h'8
                # addr:'h'b
                # addr:'h'c
                # addr:'h'd
                # addr:'h'e;
```

```
seg_e.d         = addr:'h'0
                # addr:'h'2
                # addr:'h'6
                # addr:'h'8
                # addr:'h'a
                # addr:'h'b
                # addr:'h'c
                # addr:'h'd
                # addr:'h'e
                # addr:'h'f;

seg_f.d         = addr:'h'0
                # addr:'h'4
                # addr:'h'5
                # addr:'h'6
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'b
                # addr:'h'c
                # addr:'h'e
                # addr:'h'f;

seg_g.d         = addr:'h'2
                # addr:'h'3
                # addr:'h'4
                # addr:'h'5
                # addr:'h'6
                # addr:'h'8
                # addr:'h'9
                # addr:'h'a
                # addr:'h'b
                # addr:'h'd
                # addr:'h'e
                # addr:'h'f;

/** Enables **/

seg_a.oe = 'b'1;
seg_b.oe = 'b'1;
seg_c.oe = 'b'1;
seg_d.oe = 'b'1;
seg_e.oe = 'b'1;
seg_f.oe = 'b'1;
seg_g.oe = 'b'1;

/** Resets **/

seg_a.ar = 'b'0;
seg_b.ar = 'b'0;
```

```
seg_c.ar = 'b'0;
seg_d.ar = 'b'0;
seg_e.ar = 'b'0;
seg_f.ar = 'b'0;
seg_g.ar = 'b'0;

/** Presets **/

seg_a.sp = 'b'0;
seg_b.sp = 'b'0;
seg_c.sp = 'b'0;
seg_d.sp = 'b'0;
seg_e.sp = 'b'0;
seg_f.sp = 'b'0;
seg_g.sp = 'b'0;
```


How to reach us:

USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, Toshikatsu Otsuki,
6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–3521–8315

MFAX: RMFAX0@email.sps.mot.com –TOUCHTONE (602) 244–6609
INTERNET: http://Design–NET.com

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298