# Dragon12 LCD Display

- The Dragon12 board has a 16 character x 2 line display
- Each character is a 5 x 7 bit matrix
- A controller chip (Hitachi HD44780) converts ASCII characters to 5x7 bit image
- The controller chip is connectred to Port K of the MC9S12
  - Bit 0 of Port K (PK0) selects command (1) or data (0)
  - Bit 1 of Port K (PK0) enables the data transfer
  - Bits 5 through 2 Port K (PK5-2) contain the data
  - Bit 7 of Port K (PK7) can be used to select read or write. The LCD on the Dragon12 board is set up for write only; you need to cut a trace to be able to read from the LCD.
- Use of the display is discussed in the `Hatronix_LCD2.pdf` datasheet which is on the CD-ROM which came with the Dragon12 board.
- You can send commands or data to the controller chip to control the LCD display
- These commands and data are detailed in the `Hatronix_LCD2.pdf` datasheet
- The commands are:
  - Clear display and return cursor to home (upper left)
  - Cursor home (don't clear display)
  - Entry mode (move cursor left or write)
  - Display on/off – turns display on or off, cursor on or off, cursor blink
  - Cursor/display shift – move cursor or shift display, which direction
  - Function set – bus data width (8 or 4), number of display lines (1 or 2), font size (6x8 or 5x7)
  - Set CG RAM address – set address of CG (Character Generation) RAM to generate your own charaters
  - Set DD RAM Address – set address of DD (Data Display) RAM to display characters
  - Read busy flag and address (we can't use)
  - Write data to DD RAM or CG RAM
  - Read data from DD RAM or CG RAM (we can't use)

## Dragon12 LCD Display

- The LCD display can use either 8-bit or 4-bit data bus. The Dragon12 board uses a 4-bit bus, so it takes two transfers to send one command
- The Dragon12 board is set so that you cannot read from the display; you can only write to it.
- When you write a command, you need to wait until the command has be executed by the LCD controller. The Busy Flag (from Read Busy Flag command) tells when the command is done. We cannot read Busy Flag, so we have to wait specified time before proceeding.
- To write to the controller, need to:
  1. Set RS (PK0) to 0 for command, 1 for data
  2. Set R/$\overline{\text{W}}$ (On Dragon12, R/$\overline{\text{W}}$ tied low for write only)
  3. Put 4 MSB on Port K bits 5-2
  4. Bring E (PK1) high for at least 230 ns
  5. Bring E (PK1) low
  6. Put 4 LSB on Port K bits 5-2
  7. Bring E (PK1) high for at least 230 ns
  8. Bring E (PK1) low
  9. Wait specified amount of time for execution to complete

## Dragon12 LCD Display

- To use LCD display
  1. Give command 0x28: Tell controller our display uses 4-bit data, 2-line display, 5x7 font
  2. Give command 0x0F: Turn on display, use cursor, blink cursor
  3. Give command 0x06: Move cursor to right after writing a character
  4. Give command 0x01: Clear screen, move cursor to home (upper left character)
  5. Wait for at least 1.64 ms
- After display is set up, you can write characters to display

File lcd.h:

```
#define  LCD_DAT  PORTK    /* Port K drives LCD data pins, E, and RS */
#define  LCD_DIR  DDRK     /* Direction of LCD port */
#define  LCD_E    0x02     /* LCD E signal */
#define  LCD_RS   0x01     /* LCD Register Select signal */

#define  CMD      0        /* Command type for put2lcd */
#define  DATA     1        /* Data type for put2lcd */

/* Prototypes for functions in lcd.c */
void openlcd(void);               /* Initialize LCD display */
void put2lcd(char c, char type);  /* Write command or data to LCD controller */
void puts2lcd (char *ptr);        /* Write a string to the LCD display */
void delay_50us(int n);           /* Delay n 50 microsecond intervals */
void delay_1ms(int n);            /* Delay n 1 millisecond intervals */
```

File lcd.c:

```
#include "derivative.h"
#include "lcd.h"

void openlcd(void)
{
     LCD_DIR = 0xFF;       /* configure LCD_DAT port for output */
     delay_1ms(100);       /* Wait for LCD to be ready */
     put2lcd(0x28,CMD);        /* set 4-bit data, 2-line display, 5x7 font */
     put2lcd(0x0F,CMD);        /* turn on display, cursor, blinking */
     put2lcd(0x06,CMD);        /* move cursor right */
     put2lcd(0x01,CMD);        /* clear screen, move cursor to home */
     delay_1ms(2);        /* wait until "clear display" command is complete */
}

void puts2lcd (char *ptr)
{
     while (*ptr) {            /* While character to send */
       put2lcd(*ptr,DATA);     /* Write data to LCD */
       delay_50us(1);          /* Wait for data to be written */
       ptr++;                  /* Go to next character */
     }
}

void put2lcd(char c, char type)
{
     char c_lo, c_hi;

     c_hi = (c & 0xF0) >> 2;             /* Upper 4 bits of c */
     c_lo = (c & 0x0F) << 2;             /* Lower 4 bits of c */
     if (type == DATA) LCD_DAT |= LCD_RS; /* select LCD data register */
     else LCD_DAT &= (~LCD_RS);          /* select LCD command register */
     if (type == DATA)
         LCD_DAT = c_hi|LCD_E|LCD_RS;    /* output upper 4 bits, E, RS high */
     else
         LCD_DAT = c_hi|LCD_E;           /* output upper 4 bits, E, RS low */
     LCD_DAT |= LCD_E;     /* pull E signal to high */
     __asm(nop);           /* Lengthen E */
     __asm(nop);
     __asm(nop);
     LCD_DAT &= (~LCD_E);  /* pull E to low */
     if (type == DATA)
         LCD_DAT = c_lo|LCD_E|LCD_RS;    /* output lower 4 bits, E, RS high */
     else
         LCD_DAT = c_lo|LCD_E;           /* output lower 4 bits, E, RS low */
```

```
     LCD_DAT |= LCD_E;     /* pull E to high */
     __asm(nop);           /* Lengthen E */
     __asm(nop);
     __asm(nop);
     LCD_DAT &= (~LCD_E);  /* pull E to low */
     delay_50us(1);        /* Wait for command to execute */
}

#define D50US 133   /* Inner loop takes 9 cycles; need 50x24 = 1200 cycles */
void delay_50us(int n)
{
    volatile int c;

    for (;n>0;n--)
        for (c=D50US;c>0;c--) ;
}

void delay_1ms(int n)
{

    for (;n>0;n--) delay_50us(200);
}
```

File main.c:

```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include "lcd.h"

void main(void) {
    char *msg1 = "Hello, World!";
    char *msg2 = "From Bill";

    openlcd();           // Initialize LCD display
    puts2lcd(msg1);      // Send first line
    put2lcd(0xC0,CMD);   // move cursor to 2nd row, 1st column
    puts2lcd(msg2);      // Send second line
    __asm(swi);
}
```