LCD Project

Richard Hoehn and Kevin Duke

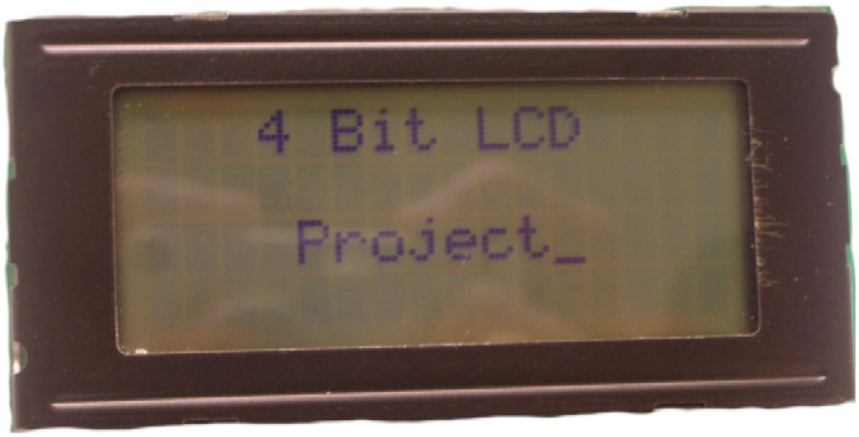
Performed: 03/09/04

Submitted: 04/29/04

Table of Contents

Abstract

Frequently, a 68HC11 program must interact with the outside world using input and output devices that communicate directly with a human being, sometime called HMI meaning Human-Machine-Interface. One of the most common devices used for HMI that can be attached to a 68HC11 is an LCD character display. Some of the most common LCDs displays connected to the 68HC11 are 16x2 and 20x2 displays. Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip, which receives data from an external source (in this case, the 68HC11) and communicates directly with the LCD panel.



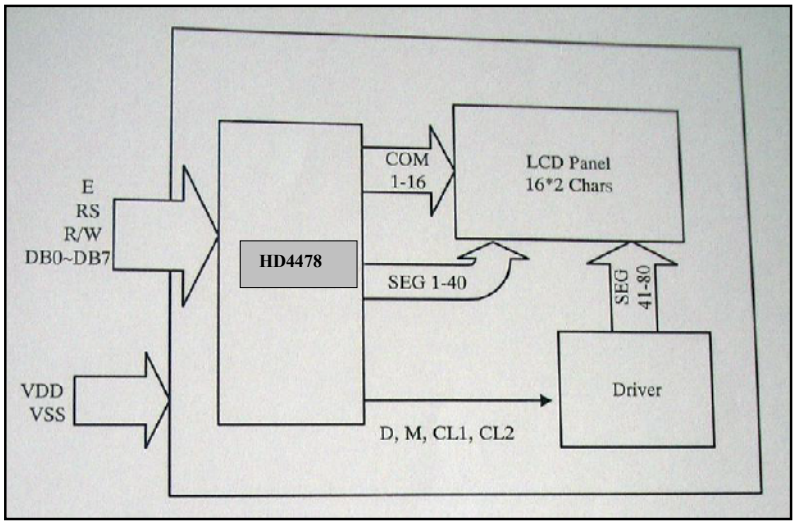
Introduction

The HD44780 standard requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used, the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7; the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus). By enabling to select what type of communications one can used lets programmers and PCB designers be flexible with how they use and interface with the LCD panels.

We will introduce in this paper how the protocols work between the Microcontroller and the LCD panel. In addition we will look into how the two different system of 4-bit and 8-bit communication differ and work.

Procedure

As we've mentioned, the LCD requires either 8 or 11 I/O lines to communicate with. We are going to use an 8-bit data bus--so we'll be using 11 of the 68HC11's I/O pins to interface with the LCD panel. The three control lines are referred to as EN, RS, RW, and are essential to correct operation of the LCD panel.



The EN or E line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should first set this line high (1) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN low (0) again. The 1-0 transition tells the HD44780 to take the data currently found on the other control lines and on the data bus and to treat it as a command.

The RS line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data, which should be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

The RW line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or

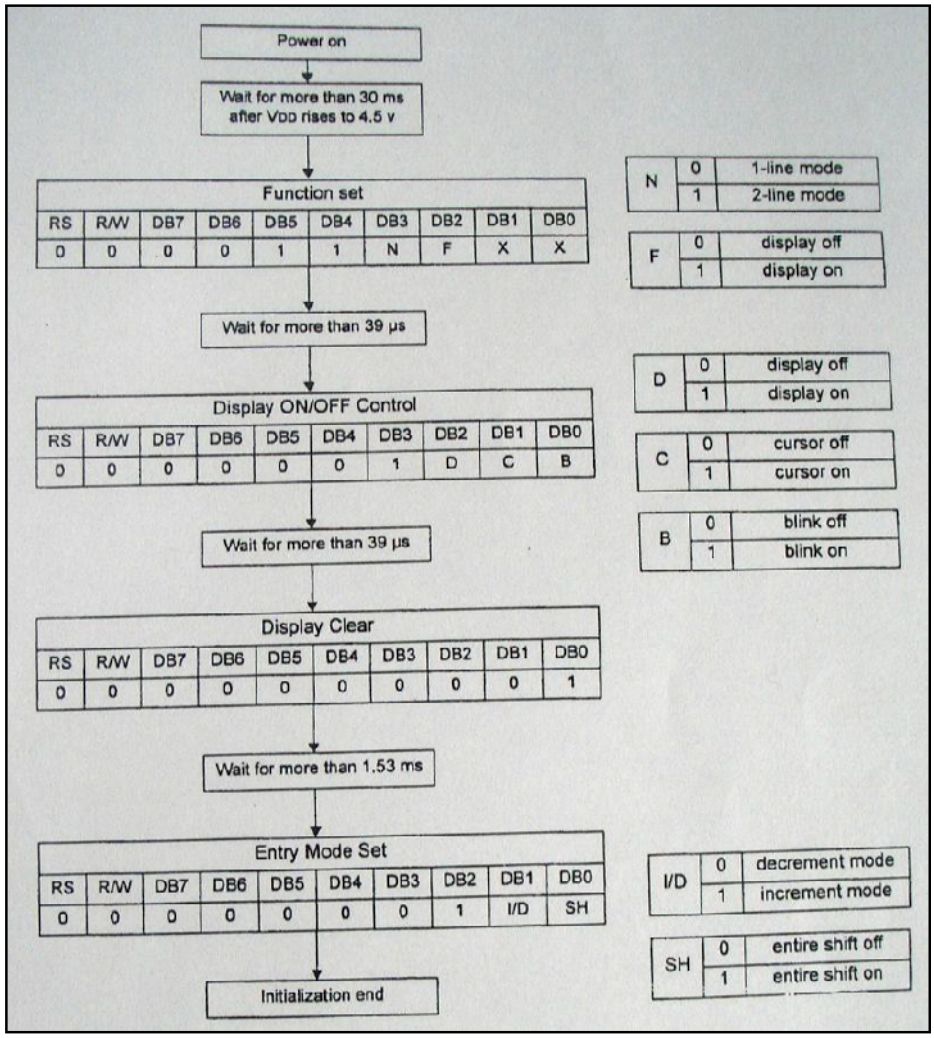
reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low.

As we mentioned above, the EN line is used to tell the LCD that you are ready for it to execute an instruction that you've prepared on the data bus and on the other control lines. Note that the EN line must be raised/lowered before/after each instruction sent to the LCD regardless of whether that instruction is read or write, text or instruction. The LCD interprets and executes our command at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed. In short, you must always manipulate EN when communicating with the LCD. EN is the LCD's way of knowing that you are talking to it. **If you don't raise/lower EN, the LCD doesn't know you're talking to it on the other lines.** Thus, before we interact in any way with the LCD we will always bring the EN line high. And once we've finished setting up our instruction with the other control lines and data bus lines, we'll always bring this line back low. Additionally, when you bring EN low and the LCD executes your instruction, it requires a certain amount of time to execute the command. The time it requires to execute an instruction depends on the instruction. So one can write a little delay to let the display update and execute the command sent to it from the microcontroller.

A more robust method of programming is to use the "Get LCD Status" command to determine whether the LCD is still busy executing the last instruction received. The "Get LCD Status" command will return to us two tidbits of information; the information that is useful to us right now is found in DB7. When we issue the "Get LCD Status" command the LCD will immediately raise DB7 if it's still busy executing a command or lower DB7 to indicate that the LCD is no longer occupied. Thus our program can query the LCD until DB7 goes low, indicating the LCD is no longer busy. At that point we are free to continue and send the next command. However this

system requires that the port is read and write enabled so it would have to be the C Port on the 68HC11 microcontroller.

Before you may really use the LCD, you must initialize and configure it. This is accomplished by sending a number of initialization instructions to the LCD, either in 8-bit or 4-bit



mode. The first instruction we send must tell the LCD whether we'll be communicating with it with an 8-bit or 4-bit data bus. We also select a 5x8 dot or a 5x7 dot character font. These two options are selected by sending the command 38h to the LCD as a command. We mentioned that the RS line must be low if we are sending a command to the LCD. The LCD command 38h is really the sum of a number of option bits. The instruction itself is the instruction 20h ("Function set"). However, to this we add the values 10h to indicate an 8-bit data bus plus 08h to indicate that the display is a two-line display. We've now sent the first byte of the initialization sequence. The second byte of the initialization sequence is the instruction 0Eh.

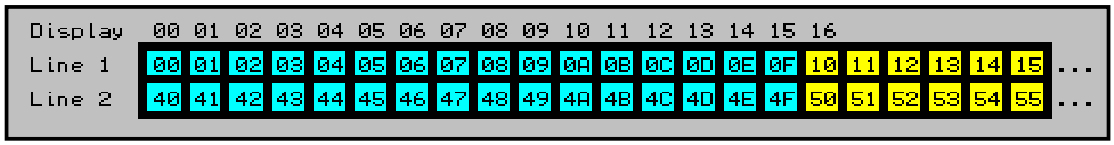
The command 0Eh is really the instruction 08h plus 04h to turn the LCD on. To that an additional 02h is added in order to turn the cursor on. The last byte we need to send is used to configure additional operational parameters of the LCD. We must send the value 06h.

The command 06h is really the instruction 04h plus 02h to configure the LCD such that every time we send it a character, the cursor position automatically moves to the right. Having executed the LCD initialization, the LCD will be fully initialized and ready for us to send display data to it.

When the LCD is first initialized, the screen should automatically be cleared. However, it's always a good idea to do things yourself so that you can be completely sure that the display is the way you want it. Thus, it's not a bad idea to clear the screen as the very first operation after the LCD has been initialized.

Now we get to the real meat of what we're trying to do: All this effort is really so we can display text on the LCD. Really, we're pretty much done.

The 44780 contains a certain amount of memory, which is assigned to the display. All the text we write to the 44780 is stored in this memory, and the 44780 subsequently reads this memory to display the text on the LCD itself. This memory can be represented with the following "memory map":



In the above memory map, the area shaded in blue is the visible display. As you can see, it measures 16 characters per line by 2 lines. The numbers in each box is the memory address that corresponds to that screen position.

Thus, the first character in the upper left-hand corner is at address 00h. The following character position (character #2 on the first line) is address 01h, etc. This continues until we reach the 16th character of the first line, which is at address 0Fh.

However, the first character of line 2, as shown in the memory map, is at address 40h. This means if we write a character to the last position of the first line and then write a second character, the second character will not appear on the second line. That is because the second character will effectively be written to address 10h--but the second line begins at address 40h.

Thus we need to send a command to the LCD that tells it to position the cursor on the second line. The "Set Cursor Position" instruction is 80h. To this we must add the address of the location where we wish to position the cursor.

Obviously we have not addresses all issues. The 44780 LCD controllers' offers many other functions, which are accessed using other commands, and some of the commands already presented include other options that were not discussed here.

We have seen that character LCD's based on the HD44780 chip can be driven in 8bits mode, which requires in total 11 lines from you microcontroller. If we want (or need) to spare some lines for other purposes it is possible to drive the display in 4bits mode, which requires 7 lines. It is possible to use only 6 lines, in which case R/W is tied to Ground. This configuration is seen many times in projects. Instead of reading the busy flag (which is somewhat trickier than it is in 8 bit modus) we have to use delay loops. The only drawback using 4 bits is that commands and data have to be sent in two nibbles (4bit parts) to the display, which take

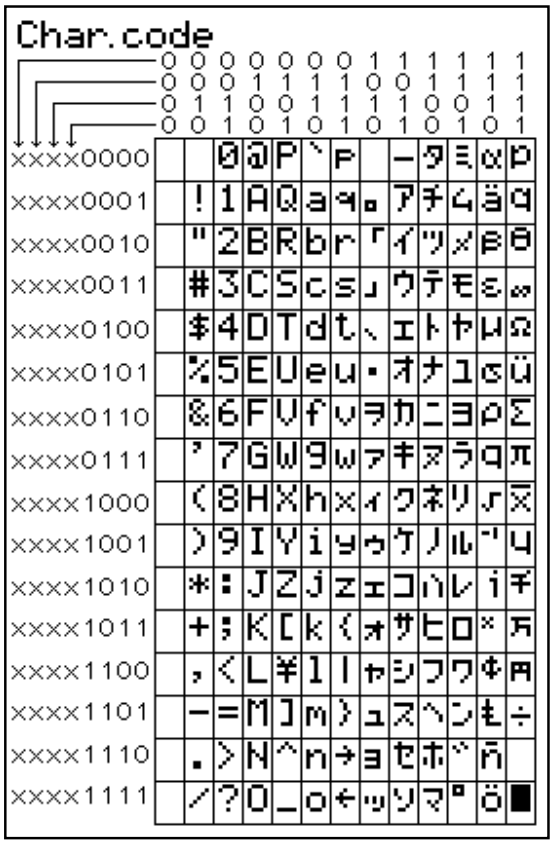
LCD PIN	SYMBOL	FUNCTION	68HC11 PIN
1	GND	Power Supply GND	GND (60)
2	VCC	Power Supply (+5V)	VCC (58)
3	LCD contrast	Contrast Adjust	71V
4	LCD RS	0 = Instruction input 1= Data Input	C(0) (9)
5	R/W	0 = Write to LCD Module 1 = Read from LCD Module	GND (59)
6	LCD ENABLE	Enable Signal	C1 (10)
7	DATA	Data bus line 0 (LSB)	B0 (42)
8	DATA	Data bus line 1	B1 (41)
9	DATA	Data bus line 2	B2 (40)
10	DATA	Data bus line 3	B3 (39)
11	DATA	Data bus line 4	B4 (38)
12	DATA	Data bus line 5	B5 (37)
13	DATA	Data bus line 6	B6 (36)
14	DATA	Data bus line 7 (MSB)	B7 (35)

*Industry standard for character LCD-modules with a maximum of 80 characters.

slightly more time. In many cases, that will not be a problem. We make two subroutines; one to read two nibbles from the LCD, and the other to write two nibbles to the LCD. Furthermore, the toggling of the EN-line is also taken to these subroutines, because we have to toggle for each nibble. To keep things simple, we will only explain the differences between 8-bit mode and 4-bit mode.

The first instruction we send must tell the LCD we'll be communicating with it with a 4-bit data bus. These two options are selected by sending the command 28h to the LCD as a command. After powering up the LCD, it is in 8-bit mode. Because only four bits are connected, the first command has to be sent twice; the first time to switch to 4-bit mode, (the lower 4 bits of the command are not sent), the second time we send it as two nibbles so the lower part is received, too.

We've now sent the first byte of the initialization sequence. The second byte of the initialization sequence is the instruction 0Eh. The last byte we need to send is used to configure additional operational parameters of the LCD. We must send the value 06h. Having executed this code the LCD will be fully initialized and ready for us to send display data to it.



To display data in 4-bit and 8-bit modes one must ensure that the RS bit is set (TRUE) so the HD44780 controller knows that we are sending data in form of characters instead of instructions to the LCD display. Next the EN should be set (TRUE) and then the data lines be set with the appropriate ASCII code for the character that one wants to display. Then you take EN low again so the controller knows that

one has sent data.

Another very nice feature of the HD44780 controller is that it has the ability to enable the microcontroller to be able to address where we want to place a certain character. However to be able to do this one has to give each character space on the LCD its own address and space. The addressing of the spaces is not quite clear however due to constraints within the HD44780 controller itself and how it addresses the LCD pixels of the displays. Because of this we have created a table with the different positions and addressing schemes for the different types of LCD character addressing.

16 characters x 1 line

First Line: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

20 characters x 1 line

First Line: 00 01 02 03 04 05 06 07 08 09 40 41 42 43 44 45 46 47 48 49

In fact, from HD44780 point of view, in this case two lines are placed one after another. So when we want to use the display from the eleventh position, it has to be initialized as if it were a TWO lines display! Mind the eleventh position is addressed as 40h, not 0Ah.

40 characters x 1 line

First Line: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11...27

The module has to be initialized as a TWO lines display, if we also want to use the second line.

24 characters x 2 lines

First Line: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
Second Line: 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57

The module has to be initialized as a TWO lines display, if we also want to use the second line.

40 characters x 2 lines

First Line: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11... 27
Second Line: 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51... 67

The module has to be initialized as a TWO lines display, if we also want to use the second line. This is also the maximum configuration, which is possible with one HD44780 + extension chips 80 characters.

16 characters x 4 lines

First Line:	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
Second Line:	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
Third Line:	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
Fourth Line:	50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F

To use the second and the fourth line, the module has to be initialized as a TWO lines display. In fact, the third line is continuous to the first line, and the fourth line is continuous two the second line from an addressing point of view.

20 characters x 4 lines

First Line:	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13
Second Line:	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53
Third Line:	14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27
Fourth Line:	54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

To use the second and the fourth line, the module has to be initialized as a TWO lines display. In fact, the third line is continuous to the first line, and the fourth line is continuous two the second line from an addressing point of view.

24 characters x 4 lines

First Line:	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
Second Line:	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37
Third Line:	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57
Fourth Line:	60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77

To use the THIRD and the FOURTH line, the module has to be initialized as a TWO lines display. Look out! There is a small 'view'-gap between the addressing of the first and the second line (and the third and fourth line respectively).

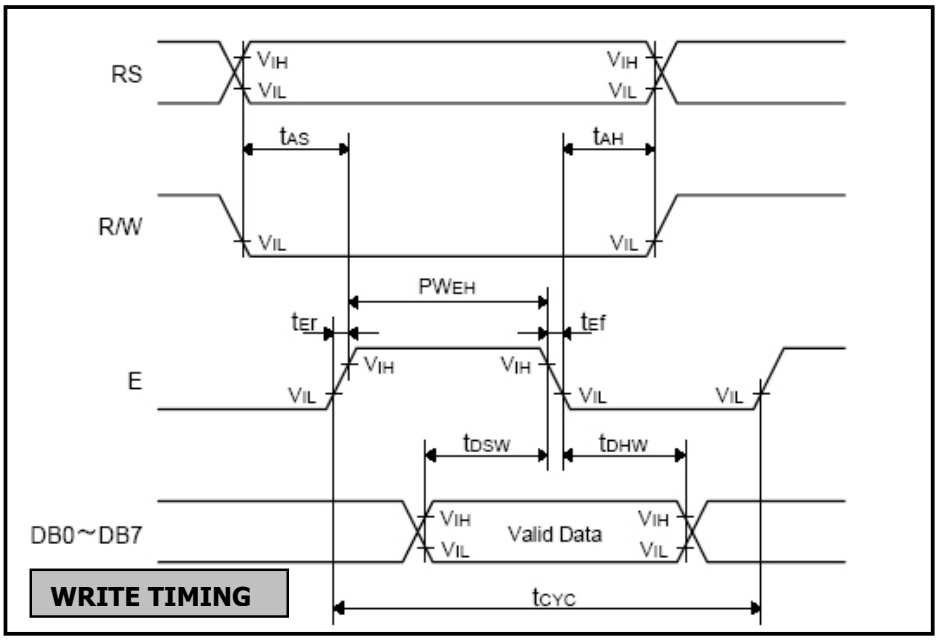
40 characters x 4 lines

First Line:	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12... 27
Second Line:	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52... 67
Third Line:	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12... 27
Fourth Line:	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52... 67

These modules uses two HD44780's (each with expansion chips) and can be seen as two 40 x 2 modules in one. All wiring is common, except for the EN (enable) lines, which are separate to drive each HD44780 apart.

Results

One of the most critical aspects of the LCD display was the timing. It is necessary to ensure correct timing. It is therefore important that one understands the inner workings of the timing for the correct communication between the 68HC11 processor and the HD44780 controller. In both 8-bit and 4-bit the timing is equally important to ensure correct operation. As mentioned in the paper above it is wise either to use a delay or the busy flag to ensure when to continue with the sending new commands to the LCD display.



Another important fact to the interfacing of the LCDs is that contrast voltage connections. Most 95% of LCDs will reference their Vee to the Vcc. When you look at the data sheets the Vee requirement is often specified as a negative voltage number with respect to Vcc. If the specifications say that Vee needs to be -5V then for a display operating at a Vcc of 5 volts it is sufficient to attach the Vee to GND. However this is most often quite confusing for new comers since they think they have to create a negative voltage of around 5VDC, which is not the case really.

Code

```
CONTROL EQU  $1003
DDRC    EQU  $1007
DATA    EQU  $1004

        ORG  $0100
MAIN
        LDAA  #$FF
        STAA  DDRC

        LDAA  #%00000010
        STAA  CONTROL
        LDAA  #$3C
        STAA  DATA
        LDAA  #%00000000
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000010
        STAA  CONTROL
        LDAA  #$0F
        STAA  DATA
        LDAA  #%00000000
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000010
        STAA  CONTROL
        LDAA  #$01
        STAA  DATA
        LDAA  #%00000000
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000010
        STAA  CONTROL
        LDAA  #$06
        STAA  DATA
        LDAA  #%00000000
        STAA  CONTROL
        JSR   DLY10MS


WRDATA:
        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #$80
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'I'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
```

```
        LDAA  #'''
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'M'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #' '
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'D'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'O'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'N'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS

        LDAA  #%00000011
        STAA  CONTROL
        LDAA  #'E'
        STAA  DATA
        LDAA  #%00000001
        STAA  CONTROL
        JSR   DLY10MS
        SWI

DLY10MS
        PSHX
        LDX   #$0D06
DLYLP   DEX
        BNE   DLYLP
        PULX
```