Lab 9 - LCD and Keyboard interfacing

Objectives:

- Practicing HCS12 Assembly.
- Learning the use of different hardware capabilities of the Dragon EVB.

Additional reference: HCS12 Microcontroller and Embedded Systems, Mazidi & Causey, chapter 12, PrenticeHall, 2008.

1. LCD

A liquid crystal display (LCD) can be a much better man-machine interface that seven segment displays. To make it easier to interface to it, Hitachi developed a module (HD44780) which comes with a local controller, allowing us to just send instructions and data to be displayed in a similar manner that we access memory chips, which became a de facto standard followed by other manufacturers. LCD became widely used in equipments with embedded controllers, like microwave ovens, for example. The datasheet for the Hitachi module can be found at:
https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Interfacing

The module can be found in several configurations of 1, 2 or 4 lines, with variable length each line, but the most commonly used is the 2x16 (2 lines of 16 characters). You can buy different version, European, Japanese or custom font version, with different sets of characters, but even the European has some configurable characters that you can program yourself, like your company logo. Figure 1 shows the ROM code A00 (Japanese) and ROM code A02 (European) character sets.

To display characters you have to send an instruction to configure it and later send the corresponding ASCII code of the character to be displayed.

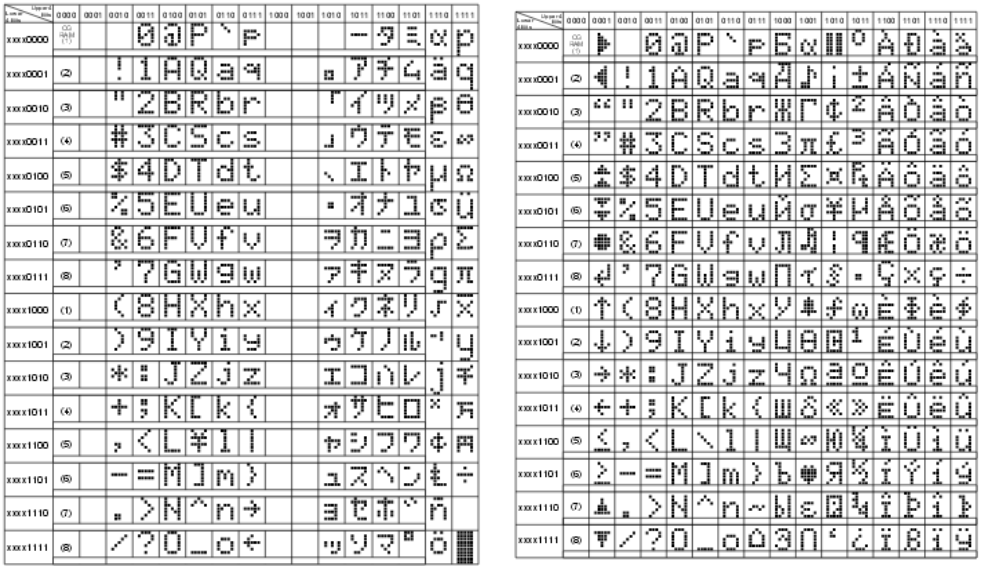


Fig.1 Japanese and European character sets

Table 7.4 Pin assignment for displays with more than 80 characters

Pin No.	symbol	I/O	Function
1	DB7	I/O	Data bus line 7
2	DB6	I/O	Data bus line 6
3	DB5	I/O	Data bus line 5
4	DB4	I/O	Data bus line 4
5	DB3	I/O	Data bus line 3
6	DB2	I/O	Data bus line 2
7	DB1	I/O	Data bus line 1
8	DB0	I/O	Data bus line 0
9	E1	I	Enable signal row 0 and 1
10	R/$\overline{W}$	I	0 = write to LCD, 1 = read from LCD
11	RS	I	0 = instruction input, 1 = data input
12	V_{EE}	-	Contrast adjust
13	V_{SS}	-	Power supply (GND)
14	V_{CC}	-	Power supply (+5 V)
15	E2	I	Enable signal row 2 and 3
16	N.C	-	

Table 7.3 Pin assignment for displays with less than 80 characters

Pin No.	symbol	I/O	Function
1	V_{SS}	-	Power supply (GND)
2	V_{CC}	-	Power supply (+5 V)
3	V_{EE}	-	Contrast adjust
4	RS	I	0 = instruction input, 1 = data input
5	R/$\overline{W}$	I	0 = write to LCD, 1 = read from LCD
6	E	I	Enable signal
7	DB0	I/O	Data bus line 0
8	DB1	I/O	Data bus line 1
9	DB2	I/O	Data bus line 2
10	DB3	I/O	Data bus line 3
11	DB4	I/O	Data bus line 4
12	DB5	I/O	Data bus line 5
13	DB6	I/O	Data bus line 6
14	DB7	I/O	Data bus line 7

The LCD pinout is in the Tables 7.3 and 7.4 from Hwang's book, the first is more common. The V_{EE} pin is used to adjust the contrast of the display, Vss and Vcc are power supply, RS selects if we will send a character or an instruction (for example, clear the display or position the cursor).

The enable signal has to receive a falling edge after a few ms in order to the LCD accept the data sent. This time can be set larger than the minimum needed, or we can monitor the bit 7 the R/W=1 (read from the LCD), but normally the first approach is used, and this pin is grounded to avoid using another microcontroller pin.

Note that it can be interfaced in two main ways using 8 bits or 4 bits, in which case you have to send the character in 2 steps:

- The 8 bit interface is used normally when you access the LCD using memory mapped IO, i.e., by reserving a space in the memory address for the peripheral. It is preferred when we have external memory and already use 2 ports for the address and data lines to acess the memory.
- The 4-bit approach is used when we don't need external memory, in smaller applications, and we use a separate port for the interface, so it uses 4 pins for the data, and pins for RS, E and R/W, although R/W is normally grounded, as mentioned before. Dragon12 board used in the lab uses the second approach using pins or port K for it:
    - PK0 (output) Pin 8 RS of LCD module
    - PK1 (output) Pin 7 EN of LCD module
    - PK2 Pin 6 DB4 of LCD module (bi-directional)
    - PK3 Pin 5 DB5 of LCD module (bi-directional)
    - PK4 Pin 20 DB6 of LCD module (bi-directional)
    - PK5 Pin 19 DB7 of LCD module (bi-directional)
    - PK7 (output) Pin 108 R/W of LCD module

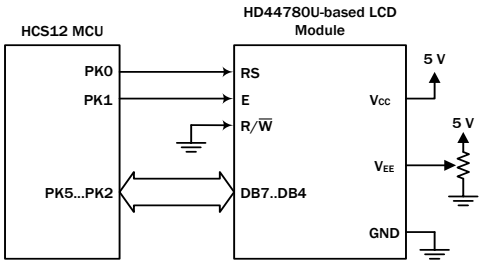


Figure 7.28 LCD interface example (4-bit bus, used in Dragon12)

You can get the board manual at
http://www.evbplus.com/download_hcs12/dragon12_plus_usb_9s12_manual.pdf

Other important facts:

- The HD44780 has a display data RAM (DDRAM) to store data to be displayed on the LCD.
- The usual way to set the display is blink the cursor, and every time you send a character, it moves to the next address.
- The address range of DDRAM for 1-line, 2-line, and 4-line LCDs are shown in Table 7.7a, 7.7b, and 7.7c.
- The HD44780 has a character generator ROM that can generates 5 × 8 or 5 × 10 character patterns from a 8-bit code.
- The user can rewrite character patterns into the character generator RAM (CGRAM). Up to eight 5 × 8 patterns or four 5 × 10 patterns can be programmed.

Table 7.7a DDRAM address usage for a 1-line LCD

Display Size	Visible	
	Character Positions	DDRAM Addresses
1 * 8	00..07	0x00..0x07
1 * 16	00..15	0x00..0x0F
1 * 20	00..19	0x00..0x13
1 * 24	00..23	0x00..0x17
1 * 32	00..31	0x00..0x1F
1 * 40	00..39	0x00..0x27

Table 7.7b DDRAM address usage for a 2-line LCD

Display Size	Visible	
	Character Positions	DDRAM Addresses
2 * 16	00..15	0x00..0x0F + 0x40..0x4F
2 * 20	00..19	0x00..0x13 + 0x40..0x53
2 * 24	00..23	0x00..0x17 + 0x40..0x57
2 * 32	00..31	0x00..0x1F + 0x40..0x5F
2 * 40	00..39	0x00..0x27 + 0x40..0x67

Table 7.7c DDRAM address usage for a 4-line LCD

Display Size	Visible	
	Character Positions	DDRAM Addresses
4 * 16	00..15	0x00..0x0F + 0x40..0x4F + 0x14..0x23 + 0x54..0x63
4 * 20	00..19	0x00..0x13 + 0x40..0x53 + 0x14..0x27 + 0x54..0x67
4 * 40	00..39 on 1st controller and 00..39 on 2nd controller	0x00..0x27 + 0x40..0x67 on 1st controller and 0x00..0x27 + 0x40..0x67 on 2nd controller

Registers of HD44780

- The HD44780 has two 8-bit user accessible registers: instruction register (IR) and data register (DR).
- To write data into display data RAM or character generator RAM, the MCU (microcontroller unit) writes into the DR register.
- The address of the data RAM should be set up with an previous instruction.
- The DR register is also used for data storage when reading data from DDRAM or CGRAM.
- The register selection is shown in Table 7.8.
- The HD44780 has a busy flag that is output from the DB7 pin (only when R/W=1, as mentioned).
- The HD44780 uses a **7-bit address counter** to keep track of the address of the next DDRAM or CGRAM location to be accessed.

Table 7.8 Register selection

RS	R/$\overline{W}$	Operation
0	0	IR write as an internal operation (display clear, etc.).
0	1	Read busy flag (DB7) and address counter (DB0 to DB6).
1	0	DR write as an internal operation (DR to DDRAM or CGRAM).
1	1	DR read as an internal operation (DDRAM or CGRAM to DR).

Timing diagrams for interfacing the LCD

Similar to memory interfacing, there are strict timing requirements to acess the LCD, shown inf Figures 7.29 and 7.30 from Hwang's book.

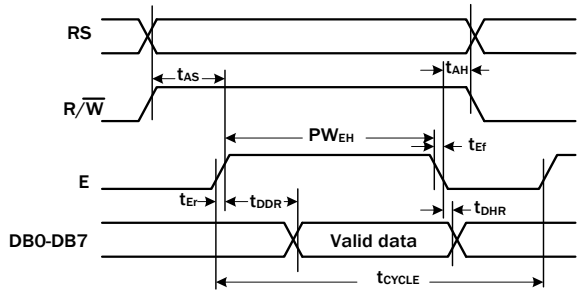


Figure 7.29 HD44780U LCD controller read timing diagram

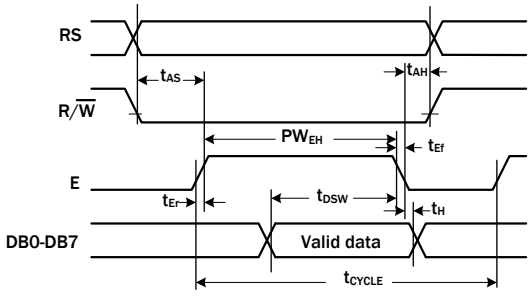


Figure 7.30 HD44780U LCD controller write timing diagram

Table 7.11 HD44780U bus timing parameters (2 MHz operation)

Symbol	Meaning	Min	Typ	Max.	Unit
t_{CYCLE}	Enable cycle time	500	-	-	ns
PW_{EH}	Enable pulse width (high level)	230	-	-	ns
t_{Er}, t_{Ef}	Enable rise and decay time	-	-	20	ns
t_{AS}	Address setup time, RS, R/$\overline{W}$, E	40	-	-	ns
t_{DDR}	Data delay time	-	-	160	ns
t_{DSW}	Data setup time	80	-	-	ns
t_H	Data hold time (write)	10	-	-	ns
t_{DHR}	Data hold time (read)	5	-	-	ns
t_{AH}	Address hold time	10	-	-	ns

SENDING an instruction to the LCD

1. Pull the RS and the E signals to low.
2. Pull the R/W signal to low.
3. Pull the E signal to high
4. Output data to the output port attached to the LCD data bus.
5. Pull the E signal to low and make sure that the internal operation is complete.

SENDING a character to the LCD

1. Pull the RS and the E signals to low.
2. Pull the R/W signal to low.
3. Pull the E signal to high
4. Output data to the output port attached to the LCD data bus.
5. Pull the E signal to low and make sure that the internal operation is complete.

-- Repeated once more for an LCD kit with 4-bit interface.

Main commands to the LCD

The datasheet form Hitachi shows several different commands, but the main ones are summarized below. The best way of imaging the data buffer is a sliding window running over the 2-line data buffer (for the 2x16 LCD).

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix (D0–D7, 8-bit)
28	2 lines and 5x7 matrix (D4–D7, 4-bit)

Remember that:

- a 2 ms delay allows the LCD to recover is needed before issuing a command or data character, we normally use a delay subroutine (does not need to be precise);
- otherwise, the busy flag can be used to see if the LCD is ready to receive information, accessed via bit D7:
    - the busy flag can be read when R/W = 1 and RS = 0
    - when D7 = 1 (busy flag = 1), the LCD is busy will not accept any new information
    - When D7 = 0, the LCD is ready

2. KEYBOARD

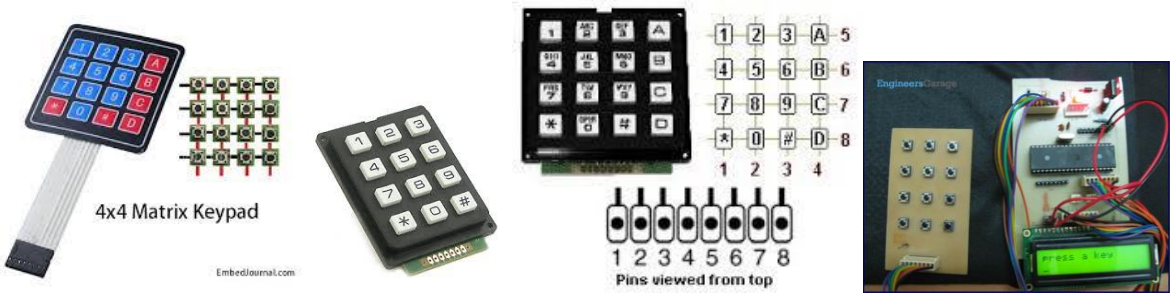


Figure 2: Examples of membrane keyboard, phone keypad, buttons arranged as keypad

Keyboards are made by grouping several switches, which can be constructed in several ways, using membrane, capacitors, hall-effect or mechanical buttons. This last one is more usual, but it generates a series of pulses because the switch contacts do not come to rest immediately. This "noise" is known as bouncing and can be interpreted as pressing the button several times. Early Coca-Cola vending machines had this problem, sometimes releasing more than one can.
Also humans cannot press faster than about 10 ms, in general if we read a key more than 50 times per second, we read the same key stroke many times.

To debounce a key, we can use three main hardware ways:

- Use a circuit similar to flip flip RS made of nands;
- Non inverting CMOS gates
- Integrated debouncer (a capacitor in parallel)

And one software way:

- Wait and see, the program waits for 10ms and check if the key is still pressed.

Interfacing to a Keyboard (from Hwang's book)
A keyboard input is divided into three steps:

1. Scan the keyboard to discover which key has been pressed.
2. Debounce the keyboard to determine if a key is indeed pressed. Both hardware and software approaches for key debouncing are available.
3. Lookup the ASCII table to find out the ASCII code of the pressed key.

ASCII Code Table Lookup
The ASCII code of each key can be stored in a table for easy look up, with the base address in an index register, and the displacement in the table comes from the key pressed.

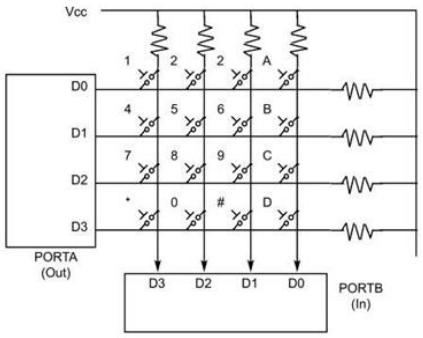
Example of look-up table

```
;This program takes a table of data, and creates a new table
; which is the original table divided by 2

PROG:   EQU   $0800      ;put program at address 0x0800
DATA:   EQU   $0900      ;put data at address 0x0900
COUNT:  EQU   10         ;number of entries in table

        ORG   PROG       ;set program counter to 0x0800
        LDX   #TABLE1    ;Reg X points to entry to process in table 1
        LDY   #TABLE2    ;Reg Y points to entry to write to table 2
        LDAB  #COUNT     ;ACC B holds number of entries left to process
REPEAT: LDAA  1,x+       ;Get table1 entry into ACC A; inc X to next entry
        ASRA             ;Divide by 2 by shifting right 1 bit
        STAA  1,y+       ;Save in table2; inc Y to next entry in table2
        DBNE  B,REPEAT   ;Decrement number left to process;
                         ;If not done, process next table1 entry
        SWI              ;Done -- Exit

        ORG DATA
         ;initialize table1 (COUNT bytes long)
TABLE1: dc.b  $07,$AE,$4A,$F3,$6C,$30,$7F,$12,$67,$CF
TABLE2: ds.b  count      ;reserve count bytes for table2.
```

Interfacing the HCS12 to a Keypad

A keypad is a smaller keyboard for simpler taks, generally with 12 (phones) to 24 kews, arranged in columns and rows. The number of pins required to interface it can be reduced by activating one columns at a time and reading if any of the columns was activated, indicating that the key in the crossing was pressed. It reduces pins, but requires more time and an algorithm to sweep all columns. A typical schematic is below, extracted from Mazidi's book – however it shows the wrong port, the Dragon board in the lab uses port A instead. The right one is from Hwang's book, and shows the correct port. The total pins required is $\log_2 Q$ divided into rows and columns, where Q is the total quantity of keys.

In larger keyboard like PCs, a dedicated microcontroller scans the keys and interfaces serially to the PC, sending the corresponding code for the key. To simplify a keypad interface, there are also ICs dedicated to scan a keyboard, as well as encoder like the 74C923.



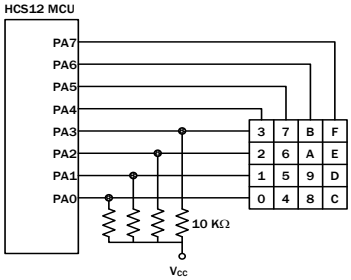


Figure 7.32 Sixteen-key keypad connected to the HCS12

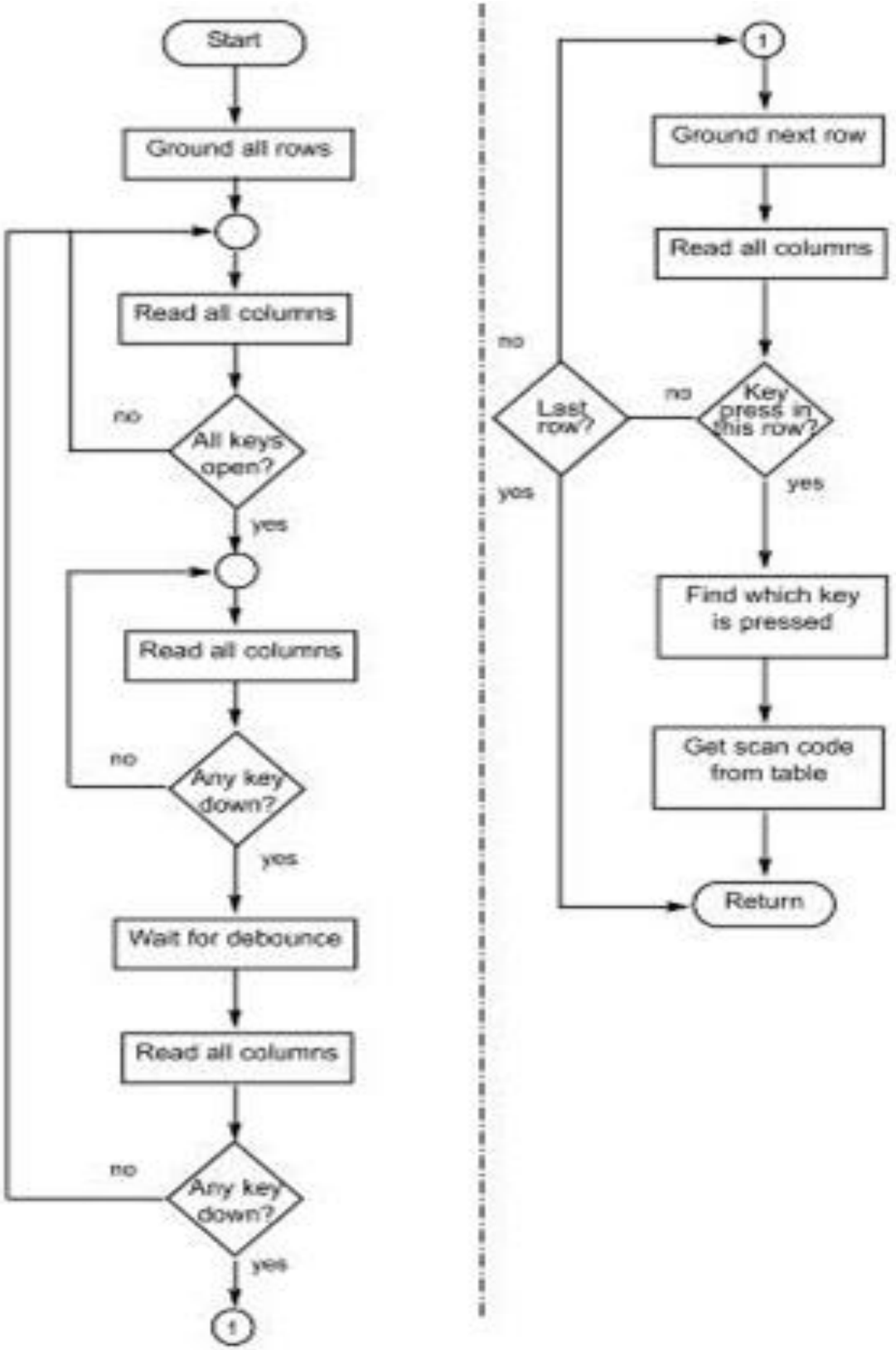
PA7	PA6	PA5	PA4	Selected Keys
1	1	1	0	0, 1, 2, and 3
1	1	0	1	4, 5, 6, and 7
1	0	1	1	8, 9, A, and B
0	1	1	1	C, D, E, and F

Table 7.12 Sixteen-key keypad row selections

Connection between HCs12 pins and the keypad at the DragonBoard

PA0 (output)	Pin 57	Col_0 of keypad	PA4 (input)	Pin 61	Row_0 of keypad
PA1 (output)	Pin 58	Col_1 of keypad	PA5 (input)	Pin 62	Row_1 of keypad
PA2 (output)	Pin 59	Col_2 of keypad	PA6 (input)	Pin 63	Row_2 of keypad
PA3 (output)	Pin 60	Col_3 of keypad	PA7 (input)	Pin 64	Row_3 of keypad

Figure 12-7 from Mazidi's book provides a flowchart for scanning and identifying the pressed key, in four stages.



In-Lab Tasks

Task 1

Assemble and execute the code provided to send a character "M" to the LCD on the Dragon Board.

Task 2

Alter the code blank the LCD and send a string with your name in the first line.

Task 3

Alter the code to blank the LCD and send a string with your name in the first line, but writing your birthday in the second line (MM/DD/YY) (hint: look up the command for cursor positioning).

Task 4

Alter the code on Task2 to use Macro instead of Subroutine as mentioned in the theory class.

Task 5

Assemble and execute the code provided to read the keyboard using lookup table – read and understand it well enough to explain its structure to someone else.

Task 6

Alter the code to read a key and send it to the LCD

Task 7

Alter the code to read a send a string of characters on the keyboard and send it to the LCD continuously.

Lab Report

All students are supposed to write their own lab reports. It should contain:

- Headers of the programs as specified in class (example in Lab08)
- Flowcharts of the code implemented
- Decode table used
- Meaningful comments in code

Note:

- TA can ask for a specific format for the report.
- TA can also ask for additional information in your report
- Lab reports are due at the beginning of your next lab session.