Advanced Parallel I/O

Dr. Abdelaziz Trabelsi
COEN 317 (Fall 2019)

Outline

- I/O addressing issue
- I/O transfer synchronization
- HCS12 parallel ports
- Electrical characteristic concerns for I/O interfacing
- Using the LCD controller

I/O Addressing Issue

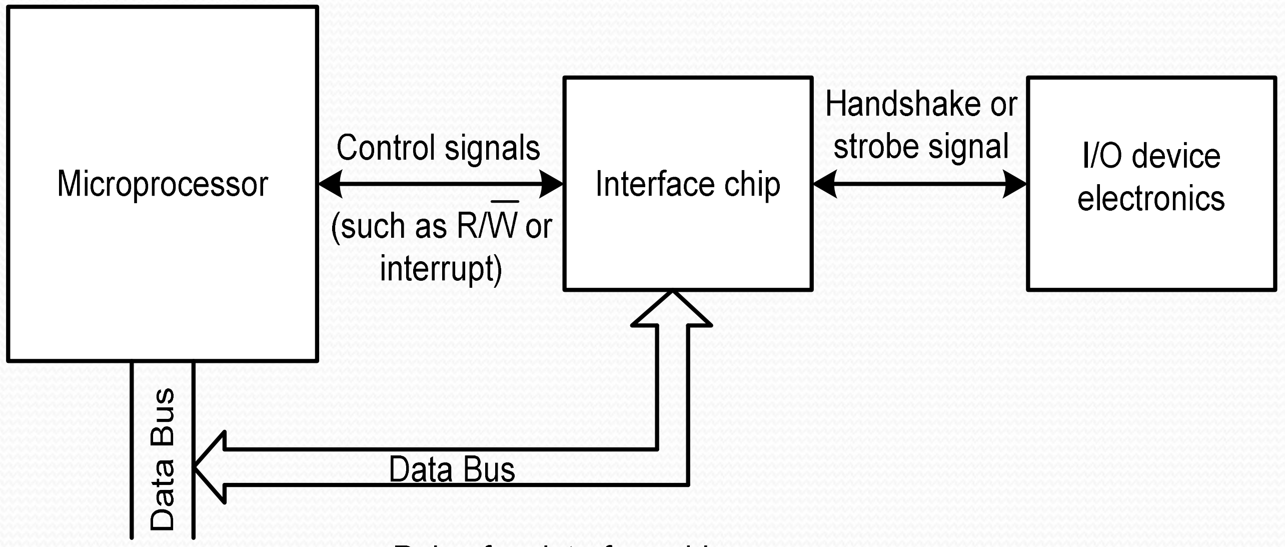
- **Parallel I/O**
  - I/O devices are also called peripheral devices as they are outside the CPU.
  - CPU needs to specify the I/O device if it needs to perform an I/O operation:
    - Specifying the address space the I/O device occupies, and
    - Making use of the instructions and addressing modes the I/O device uses.
  - In HCS12, memory devices (e.g., SRAM, EEPROM) and I/O devices occupy the same address space.

I/O Addressing Issue (Contd.)

- **I/O schemes**
  - Isolated (a.k.a. standard) I/O scheme
    - MPU has dedicated instructions for I/O operations.
    - MPU has a separate address space for I/O devices.
  - Memory-mapped I/O scheme
    - MPU uses the same instruction set to perform memory accesses and I/O operations.
    - I/O devices and memory components are resident in the same memory space.

I/O Transfer Synchronization

- MPU usually communicate with I/O devices via interface chips (or interface logic for the MCU).
- Proper communication must be carried out between:
  - MPU and interface logic, and
  - Interface logic and I/O devices.



Role of an interface chip

I/O Transfer Synchronization (Contd.)

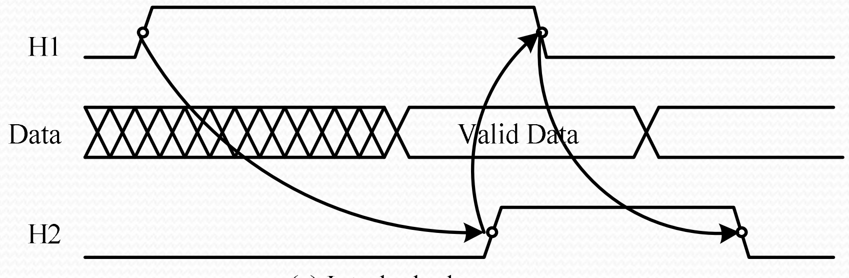
- **Synchronization issue for parallel ports**
  - Data written into the data register appears on the output pins directly.
  - Data read from the data register reflects the instantaneous voltage levels on the input pins.
  - Capability provided by most today's MCUs (including HCS12).
- **Synchronization issue for serial interfaces**
  - Synchronization is achieved by following data transfer protocols.
  - MPU needs to make sure that new data is present in the interface logic (usually held in the data register) before reading it.
  - MPU also needs to make sure that interface logic can handle more data before sending new data to interface logic.
  - Polling and interrupt methods are very useful.

I/O Transfer Synchronization (Contd.)

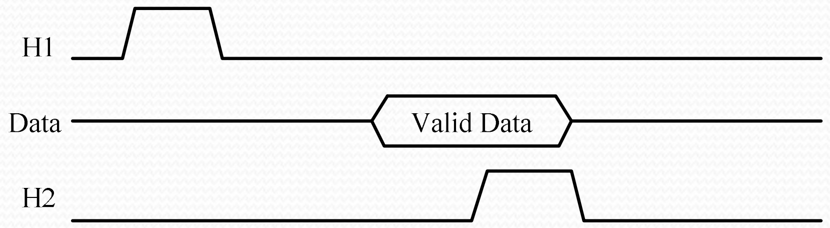
- **Polling method**
  - For input: MPU checks a status bit of the interface chip to find out if the interface chip has received new data from the input device.
  - For output: MPU checks a status bit of the interface chip to find out if it can send new data to the interface chip.
- **Interrupt-driven method**
  - For input: the interface chip interrupts the MPU whenever it has received new data from the input device.
  - For output: the interface chip interrupts the MPU whenever it can accept new data from the microprocessor.

I/O Transfer Synchronization (Contd.)

- **Input handshake protocol**
  - Step 1: interface chip asserts H1 to indicate its intention to input data.
  - Step 2: input device puts data on data port pins and asserts handshake signal H2.
  - Step 3: interface chip latches data and de-asserts H1. After some delay, the input device also de-asserts H2.



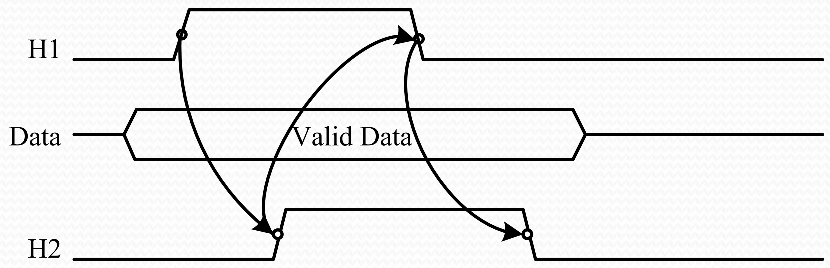
(a) Interlocked



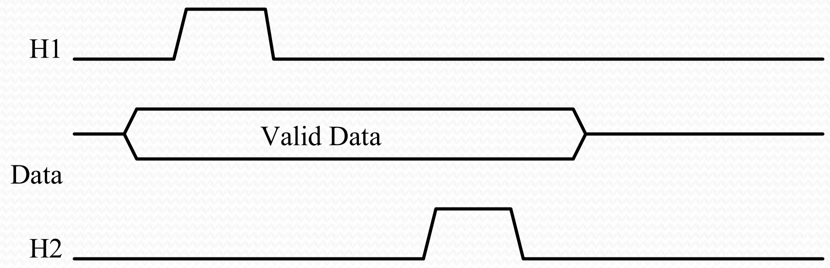
(b) Pulse mode

I/O Transfer Synchronization (Contd.)

- **Output handshake protocol**
  - Step 1: interface chip places data on port pins and asserts H1 to indicate that it has valid data to be output.
  - Step 2: output device latches the data and asserts H2 to acknowledge the receipt of data.
  - Step 3: interface chip de-asserts H1 following the assertion of H2. The output device then de-asserts H2.



(a) Interlocked



(b) Pulse Mode

HCS12 Parallel Ports

- **Device User Guide** shows the detailed register map (Section 1.6) of the MC9S12DG256 from $0000 to $03FF (1KB).
- `mc9s12dg256.inc` file contains all the numeric equivalent of the port numbers as well as memory positions.
- Normal Single-Chip Mode will be used (i.e., no external buses)
  - User configures an I/O port for input or output by programming the associated data direction register.
  - To output (‘1’), the user writes data on the port data register.
  - To input (‘0’), the user reads data form the port data register.
  - Most I/O ports have additional registers to control their operations, and most I/O pins service multiple purposes.

HCS12 Parallel Ports (Contd.)

- PORTA
  - GPIO port or data register (`PORTA`) located at $0000.
  - Connected to the keypad on the Deagon12_Light Board.
  - If the keypad is used, `PORTA` must be configured so that the bits 7 to 4 are outputs and the bits 3 to 0 are inputs (using `DDRA` register located at $0002).
- PORTB
  - GPIO port or data register (`PORTB`) located at $0001.
  - Connected to the LEDs on the Board.
  - If LEDs are used, `PORTB` must be configured so that all the bits are outputs (using `DDRB` register located at $0003).
  - PB0-PB3 are also connected to DC Motor Driver (TB6612F NG).

HCS12 Parallel Ports (Contd.)

- PORTE
  - GPIO port or data register (`PORTE`) located at $0008.
  - Pin 0 connected to `/XIRQ` and pin 1 connected to `/IRQ`.
  - `DDRE` register located at $0009.
  - Port E assignment register (`PEAR`): write $10 to configure 8 bits as I/O pins.
- PORTK
  - GPIO port or data register (`PTK`) located at $0032.
  - Connected to LCD controller on the Board.
  - If the LCD Controller is used, `PORTK` must be configured so that 4 bits are outputs and other inputs (using `DDRK` register located at $0033).
    - The LCD module operates in a write-only, 4-bits mode.
  - PK0-PK5 are connected to the alphanumeric LCD module (EL-1602A, controller compatible with HD44780).

HCS12 Parallel Ports (Contd.)

- PORTT
  - GPIO port or data register (`PTT`) located at $0240.
  - `DDRT` register located at $0242.
  - Port T pins can also be used as input capture or input capture pins on the HCS12 Timer Module.
  - Speaker is driven by PT5.
- PORTS
  - GPIO port or data register (`PTS`) located at $0248.
  - `DDRT` register located at $024A.
  - PS0 is RXD0 (SCI0 Receive), PS1 is TXD0 (SCI0 Transmit), PS2 is RXD1 (SCI1 Receive), PS3 is TXD1 (SCI1 Transmit), PS4 is MISO0 (SPI0 input), PS5 is MOSI0 (SPI0 output), PS6 is SCK0 (SPI0 clock output), and PS7 is /SS0 (SPI0 output enable).
  - The DAC (LTC1661) uses SPI to communicate via the pins PS5 (MOSI0) and PS6 (SCK0).

HCS12 Parallel Ports (Contd.)

- PORTM
  - GPIO port or data register (`PTM`) located at $0252.
  - `DDRT` register located at $0252.
  - Module routing register (`MODRR`): `$0257`. it configures the rerouting of CAN and SPI ports on defined port pins.
  - PM0 is CAN0 Receive, PM1 is CAN0 Transmit (CAN chip is the MCP2551).
  - PM2 is the common anode (in negative logic) of the RGB LED, PM6 is the Chip Select (CS) input of the DAC (LTC1661).
- PORTH
  - GPIO register located at $0260.
  - `DDRT` register located at $0252.
  - Connected to DIP Switch and Push Buttons (4 LSBs) on the Board.

HCS12 Parallel Ports (Contd.)

- PORTJ
  - GPIO port or data register (`PTJ`) located at \$0268.
  - `DDRT` register located at \$026A.
  - PJ6 is the SDA pin (I^2C) and PJ7 is the SCL pin (I^2C).
- PORTP
  - GPIO port or data register (`PTP`) located at \$0258.
  - `DDRT` register located at \$025A.
  - If PWM is enabled, all PORT P pins can become PWM channels.
  - PP0-PP3 are the cathodes of the 7-segment displays.
  - PP0-PP1 are connected to the DC Motor Driver (TB6612F NG).
  - PP4-PP6 are the anodes (in negative logic) of the RGB LED.
  - With a jumper, PP5 connects to the buzzer.

HCS12 Parallel Ports (Contd.)

- PORTs AD0 and AD1
  - Analog input interfaces to ADC subsystem.
    - Each ADC has 8 channels.
  - ATD0 Data Register (PORTAD0): $008F.
    - ATD0 Digital Input Enable Register (ATD0DIEN): $008D.
    - Each input channel can be configured as analog input (0) or digital input (1).
  - ATD1 Data Register (PORTAD1): $012F.
    - ATD1 Digital Input Enable Register (ATD1DIEN): $012D.
    - Each input channel can be configured as analog input (0) or digital input (1).
  - Some pins are connected to the light sensor, temperature sensor, Trimmer Port (see Board User Manual).
    - PAD0-PAD7: ATD0, PAD8-PAD15: ATD1?

HCS12 Parallel Ports (Contd.)

- Summary

Number of pins available in each paralle port

Port Name	No. of Pins	Pin Name
A	8	PA7~PA0
B	8	PB7~PB0
E	8	PE7~PE0
H	8	PH7~PH0
J	4	PJ7~PJ0
K	7	PK4~PK0
M	8	PM7~PM0
P	8	PP7~PP0
S	8	PS3~PS0
T	8	PT7~PT0
PAD1, PAD0	16	PAD15~PAD0
L	8	PL7~PL0
U	8	PU7~PU0
V	8	PV7~PV0
W	8	PW7~PW0

Electrical Characteristic Concerns

- **For device X to drive device Y correctly:**
  - Output high-voltage of device X (V_{OHX}) must be higher than the input high voltage of device Y (V_{IHY}).
  - Output low-voltage of device X (V_{OLX}) must be lower than the input low voltage of device Y (V_{ILY}).
  - Input and output voltage levels of several popular logic families are shown in the Table shown in the next slide.
  - At the same power supply level, CMOS device has no problem in driving bipolar and CMOS devices.
  - Bipolar devices have problem in driving CMOS devices.
  - HCS12 cannot be driven by bipolar devices with same power supply voltage.
  - Bipolar devices have problem in driving CMOS devices (including HCS12).

Electrical Characteristic Concerns (Contd.)

- In addition to MCU, most embedded systems use peripherals (or I/O devices) which usually feature an integrated circuit (IC).
- Because these ICs might use different technologies, we must make sure that they are electrically compatible:
  - Voltage-level compatibility.
  - Current drive capability.
  - Timing compatibility.

Electrical Characteristic Concerns (Contd.)

- Voltage-level compatibility
  - High output level of the IC must match that of another IC (which can be the MCU itself).
  - Same for the low output level.
- Current drive capability
  - Output of an IC might not have enough current to drive its load.
  - Total current required to drive I/O devices might exceed the maximum current rating for the MCU.
  - Solved by adding buffer chips (powered externally, such as 74ABT244) that can supply enough current between MCU and the peripheral ICs.

Electrical Characteristic Concerns (Contd.)

- Input and output voltage levels of common logic families

Logic family	VCC	V_{IH}	V_{OH}	V_{IL}	V_{OL}
HCS12[3]	5 V	3.25 V	4.2 V	1.75 V	0.8 V
S[4]	5 V	2 V	3.0~3.4 V[1]	0.8 V	0.4~0.5 V[2]
LS[4]	5 V	2 V	3.0~3.4 V[1]	0.8 V	0.4~0.5 V[2]
AS[4]	5 V	2 V	3.0~3.4 V[1]	0.8 V	0.35 V
F[4]	5 V	2 V	3.4 V	0.8 V	0.3 V
HC[3]	5 V	3.5 V	4.9 V	1.5 V	0.1 V
HCT[3]	5 V	3.5 V	4.9 V	1.5 V	0.1 V
ACT[3]	5 V	2 V	4.9 V	0.8 V	0.1 V
ABT[5]	5 V	2 V	3 V	0.8 V	0.55 V
BCT[5]	5 V	2 V	3.3 V	0.8 V	0.42 V
FCT[5]	5 V	2 V	2.4 V	0.8 V	0.55 V

Notes.

1. V_{OH} value will get lower when output current is larger.
2. V_{OL} value will get higher when output current is larger. The V_{OL} values of different logic gates are slightly different.
3. HCS12, HC, HCT, ACT are based on the CMOS technology.
4. S, LS, AS and F logic families are based on the bipolar technology.
5. ABT, BCT, and FCT are using the BiCMOS technology.

Electrical Characteristic Concerns (Contd.)

- Current capabilities of common logic families

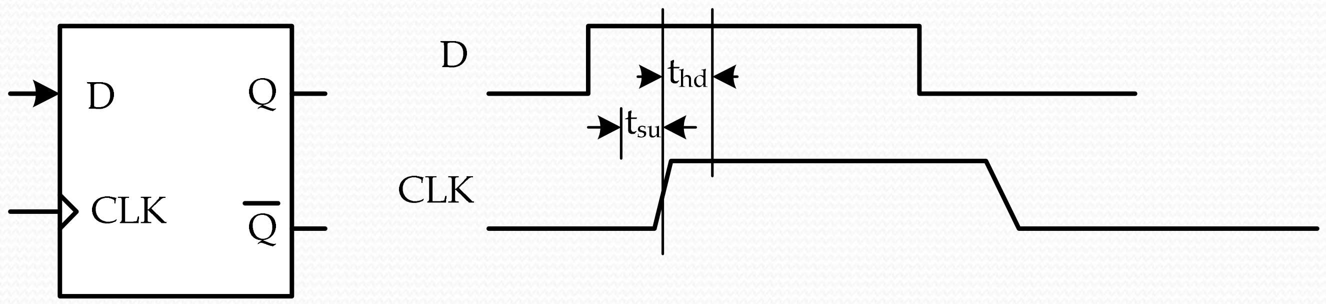
Logic family	VCC	I_{IH}	I_{IL}	I_{OH}	I_{OL}
HCS12[23]	5 V	2.5 μA	2.5 μA	25 mA	25 mA
S	5 V	50 μA	1.0 mA	1 mA	20 mA
LS	5 V	20 μA	0.2 mA	15 mA	24 mA
AS	5 V	20 μA	0.5 mA	15 mA	64 mA
F	5 V	20 μA	0.5 mA	1 mA	20 mA
HC[3]	5 V	1 μA	1 μA	25 mA	25 mA
HCT[3]	5 V	1 μA	1 μA	25 mA	25 mA
ACT[3]	5 V	1 μA	1 μA	24 mA	24 mA
ABT[3]	5 V	1 μA	1 μA	32 mA	64 MA
BCT	5 V	20 μA	1 mA	15 mA	64 mA
FCT[3]	5 V	1 μA	1 μA	15 mA	64 mA

Notes.

1. Values are based on the 74xx244 of Texas Instrument (xx is the technology name)
2. The total HCS12 supply current is 65 mA.
3. The values for I_{IH} and I_{IL} are input leakage currents.

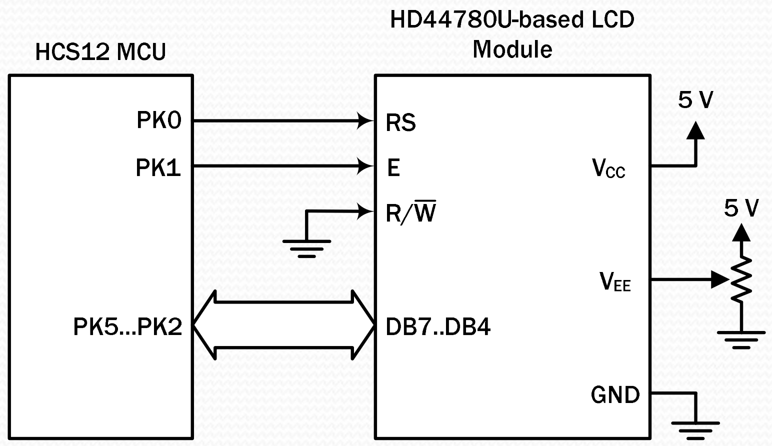
Electrical Characteristic Concerns (Contd.)

- Timing compatibility
  - An I/O pin configured as an input usually has a register that reads from a peripheral pin.
  - An I/O pin configured as an output usually has a register whose value is read by the peripheral pin (possibly on a register).
  - Must make sure not to violate the setup and hold time requirements on the flip-flops.



Using the LCD Controller

- LCD module
  - Includes the LCD and its controller.
  - The controller is compatible with the popular display controller HD44780.
  - Features of the LCD in the Dragon12-Light Board: 2 lines, 16 characters per line, only writes allowed, 4-bits mode.
  - Interface port: PORT K.



	7	6	5	4	3	2	1	0
PTK ($0032):	PTK7	PTK6	PTK5	PTK4	PTK3	PKT2	PTK1	PTK0
LCD pins:			DB7	DB6	DB5	DB4	EN	RS

Using the LCD Controller (Contd.)

- Two registers in the LCD can be accessed:
  - Data Register DR (by setting RS=1).
  - Instruction Register IR (by setting RS = 0).
- When writing to either register, enable bit must be asserted (EN=1).
- We can write an 8-bit value on both registers: DB7-DB0.
  - Due to LCD connection in Dragon12-Light Board, the 4-bits mode is used.
  - In a 4-bits mode, the 8-bit value is transmitted by first sending the higher nibble (DB7-DB4) and then sending the lower nibble (DB3-DB0).
  - Both nibbles are sent using the pins DB7-DB4.

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

Using the LCD Controller (Contd.)

- Step 1: configure the LCD by writing a set of instructions on IR.
  - Must make RS = 0 (to select the IR).
  - For a list of instructions, see Table 7.5 in the textbook.
  - Suggested list of instructions to configure the LCD:
    - Write 0x28: set 4-bit data, 2-line display, 5x8 font
    - Write 0x0F: turn on display and cursor (blink cursor position).
    - Write 0x06: set cursor move direction to the right (no display shift).
    - Write 0x80: set DDRAM address to 0 (thus, DDRAM data are accepted on DR).
    - Write 0x01: clear display, return cursor to home position (address 0, located in line 1).
    - In addition, to move the cursor to the second line:
      - Write 0xC0: set 4-bit data, 2-line display, 5x8 font.

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).

Using the LCD Controller (Contd.)

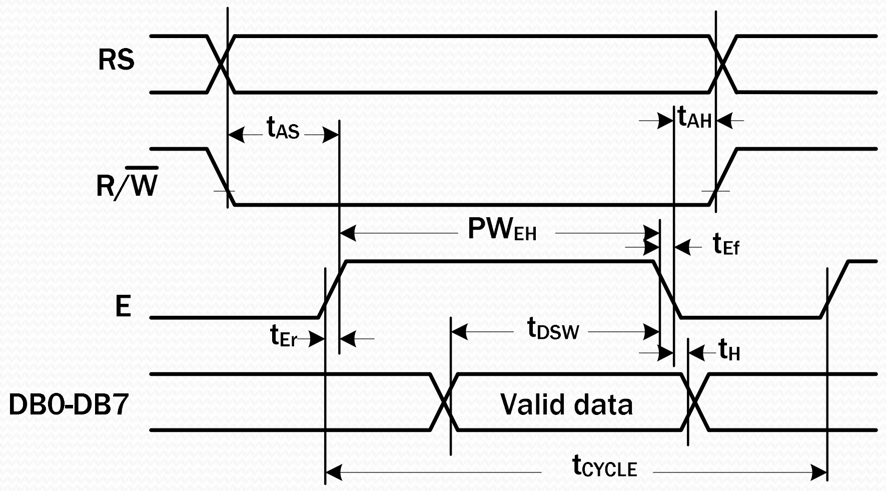
- Step 2: write ASCII characters on the LCD by writing on DR.
  - Must make RS = 1 (to select the IR).
  - Write on DDRAM or CGRAM (usually starting from address 0).
    - To write on DDRAM, first write the instruction “Set DDRAM address” on IR.
    - To write on CGRAM, first write the instruction “Set CGRAM address” on IR.
    - For a 2x16 display size, character positions and display addresses are provided in Table 7.7b in textbook.
  - HC44780 uses a counter to keep track of the address of next location to be accessed.
    - When “Set DDRAM address” or “Set CGRAM address” instructions are written on IR, the starting address is transferred to this counter.
    - Then, every write to either memory will increment (cursor moves to the right) or decrement (cursor moves to the left) automatically.
    - Whether to increment or decrement depends on instruction “Entry Mode Set”.

Using the LCD Controller (Contd.)

- Example

  Writing a byte on instruction register or data register.

  - Must comply with the following timing diagram:



HD44780U LCD controller write timing diagram.

Provided **C Code:** unit8a.c   mixasm_8a.h

Provided **ASM Code:** mixasm_8a.asm

Using the LCD Controller (Contd.)

- Example

  Writing a byte on instruction register or data register.

Writing on IR. RS = 0	**Writing on DR. RS = 1**
- Set RS = 0, E = 0; wait 60 ns - Set E=1 - Send higher nibble; wait 80 ns - Set RS = 0, E = 0; wait 60 ns - Set E=1 - Send lower nibble; wait 80 ns - Make RS = 0, E = 0 - Wait 50 us (wait time for order to be completed, true for most instructions)	- Set RS = 1, E = 0; wait 60 ns - Set E=1 - Send higher nibble; wait 80 ns - Set RS = 1, E = 0; wait 60 ns - Set E=1 - Send lower nibble; wait 80 ns - Make RS = 1, E = 0 - Wait 50 us (wait time for order to be completed, true for most instructions)

`Provided` **`C Code:`** `unit8a.c   mixasm_8a.h`

`Provided` **`ASM Code:`** `mixasm_8a.asm`

- `openLCD()`: configures the LCD.
- `putcLCD(char cx)`: places a byte on DR (ASCII characters are accepted).
- `putsLCD(char *ptr)`: places a character string on the LCD.