Real Time System

Third Level

I/O Interfacing and Programmable Devices

Dr. Hussein AbdulAmeer

Hussein.Alkhamees@uomus.edu.iq

Goals
Up-on completing this lecture, the student should be able to:

1- Identify the concepts behind interfacing using octal buffers
2- Comprehend the c/c of 74LS373, 74LS374.

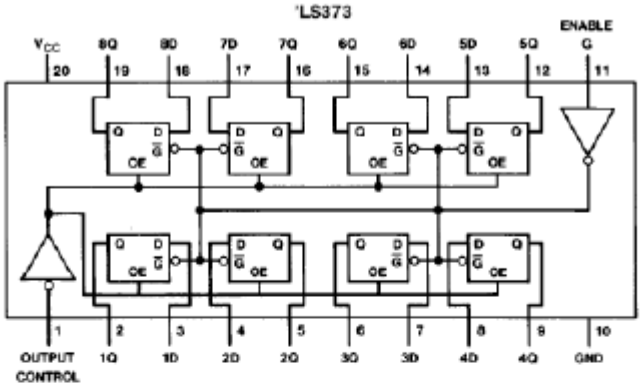
74LS373 / 74LS374 3-State Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

The 74LS37X (3, 4) consists of eight registers (latches, flip flop) with 3-state outputs. The eight latches of the 74LS373 are D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was set up. The eight flip-flops of the 74LS374 are edge-triggered D-type flip flops. On the positive transition of the clock, the Q outputs will be set to the logic states that were set up at the D inputs. The 74LS37X (3, 4) is a high-speed, low-power Octal D-type (latch or Flip-Flop) featuring separate D-type inputs for each latch or flip-flop and 3-state outputs. A buffered Clock and Output Enable is common to all latches and flip-flops. The 74LS37X (3, 4) is manufactured using advanced Low Power technology and is compatible with all TTL families. A buffered output control input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or a high-impedance state.

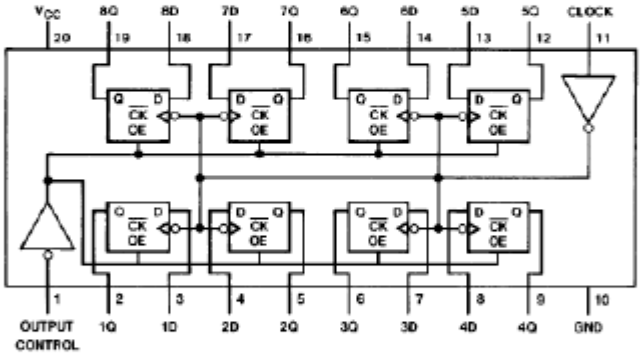
Connection Diagrams

Dual-In-Line Packages

'LS373



'LS374



Function Tables

DM54/74LS373

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

H = High Level (Steady State), L = Low Level (Steady State), X = Don't Care
↑ = Transition from low-to-high level, Z = High Impedance State
Q_0 = The level of the output before steady-state input conditions were established.

Function Tables

DM54/74LS373

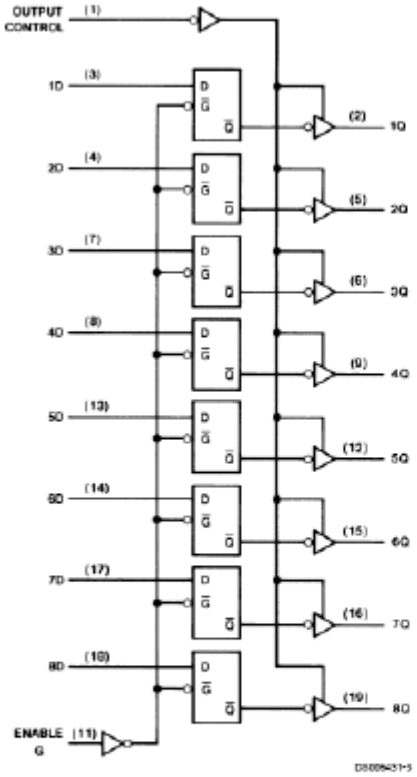
Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

H = High Level (Steady State), L = Low Level (Steady State), X = Don't Care
↑ = Transition from low-to-high level, Z = High Impedance State
Q_0 = The level of the output before steady-state input conditions were established.

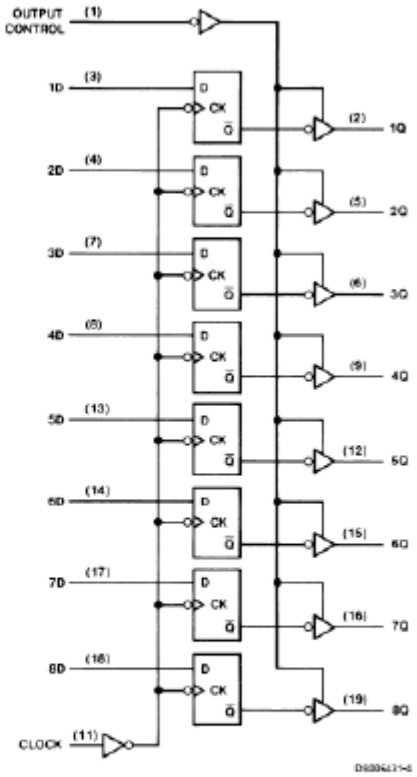
DM54/74LS374

Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

DM54/74LS334
Transparent Latches



DM54/74LS374
Positive-Edge-Triggered Flip-Flops



Absolute Maximum Ratings

- Supply Voltage 7V.
- Operating Free Air Temperature Range 0°C to +70°C.
- Storage Temperature Range -65°C to +150°C.

Absolute Maximum Ratings

- Supply Voltage 7V.
- Operating Free Air Temperature Range 0°C to +70°C.
- Storage Temperature Range -65°C to +150°C.

Recommended Operating Conditions

Symbol	Parameter	DM74LS373			Units
		Min	Nom	Max	
V_{CC}	Supply Voltage	4.75	5	5.25	V
I_{OH}	High Level Output Current			−2.6	mA
I_{OL}	Low Level Output Current			24	mA

Recommended Operating Conditions

Symbol	Parameter	DM74LS374			Units
		Min	Nom	Max	
V_{CC}	Supply Voltage	4.75	5	5.25	V
I_{OH}	High Level Output Current			−2.6	mA
I_{OL}	Low Level Output Current			24	mA

3.5.1 Interfacing to 8-bit Bus

In a simple system, a tri-state buffer such as 74LS244 can be used as an input port and a latch such as 74LS374 can be used as an output port.

Programmable Devices:-

The CPU needs to check whether a peripheral is ready before reads from or write into a device because the execution speed of the processor is much faster than the response of the peripheral such as printer. For example, when CPU sends data byte (characters) to a printer, the processor can execute the instructions to transfer a byte in (ns or μs), on the other hand, the printer can take 10 to 25 ms to print a character. After transferring a character to the printer, the CPU should wait until the printer is ready for the next character, otherwise the data will be lost. To prevent the loss of data, signals are exchanged between the CPU and peripheral before to data transfer; these signals are called handshake signals.

So we can summarize the requirements for a programmable interfacing device as follow:

The device should include:

- Input and output registers (latches and flip flops).
- Tri-state buffers.
- Bidirectional data flow.
- Handshake and interrupt signals.
- Control logic.
- Chip select logic.
- Interrupt control logic.

Is the 74LS245 programmable or not?

Can we make it programmable? How?

- The direction of data flow is determined by DIR.
- When E = 0 the 47ls245 is Bidirectional, DIR = 0 the data flow from B to A, DIR = 1 the data flow from A to B. when E = 1 it's became isolation.
- We can program the 74LS245 by CPU when we connect to register called control register and connect the DIR on the any bits of register.
- The CPU controls the 74LS245 by sending appropriate control word register to activate the proper value of DIR. this word called control word of programmable device.

Now the question is: How would the MPU write into the control register? It does so the same way it would with any other I/O port, through a port address. Figure 13.2 shows that the address lines A_7–A_1 are used to select the chip through a NAND gate and A_0 is used to differentiate between the control register and the transreceiver. When A_0 is high, the control register is enabled, and when A_0 is low, the transreceiver is enabled. Thus, the MPU could access the control register through the port address FF_H; and the transreceiver through FE_H. To set up the transreceiver as an output device, the control word would be 01_H, and to set it up as an input device the control word would be 00_H.

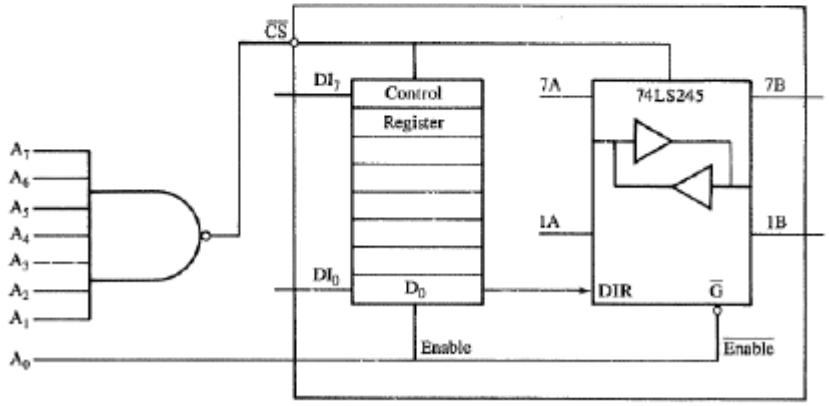


FIGURE 13.2
Making 74LS245 Programmable

Examples:

Example 13.1 Write instructions to initialize the hypothetical chip (Figure 13.2) as an output buffer and send a byte.

Solution

Instructions

```
LD A, 01H       ;Set D0 = 1, D1 through D7 are ''don't care'' lines
OUT (FFH), A    ;Write in the control register
LD A, BYTE1     ;Load data byte
OUT (FEH), A    ;Send data out
```

In the last example, we used the 74LS245 as a tri-state buffer. However, in microprocessor applications, we often need registers that can be used as I/O ports. We can build a latch with a buffer, and by controlling the enable signal of the latch, we can program it to function as an input port or an output port. Figure 13.3 shows two latches (representing eight latches in each direction); the enable signals of these latches are controlled by bit D_0 in the control register. If bit D_0 is 0, it enables the output latch, and if D_0 is 1, it enables the input latch. Thus, by programming bit D_0, we can make the device function as an input port or an output port. When the device is programmed as an output device, the MPU can write to the port by using the $\overline{WR}$ control signal to enable the tri-state buffer and to send out a byte. When bit $D_0 = 1$, the input latch is enabled and the output latch is disabled, and the MPU can read by enabling the input buffer. If we have additional registers (or I/O ports),

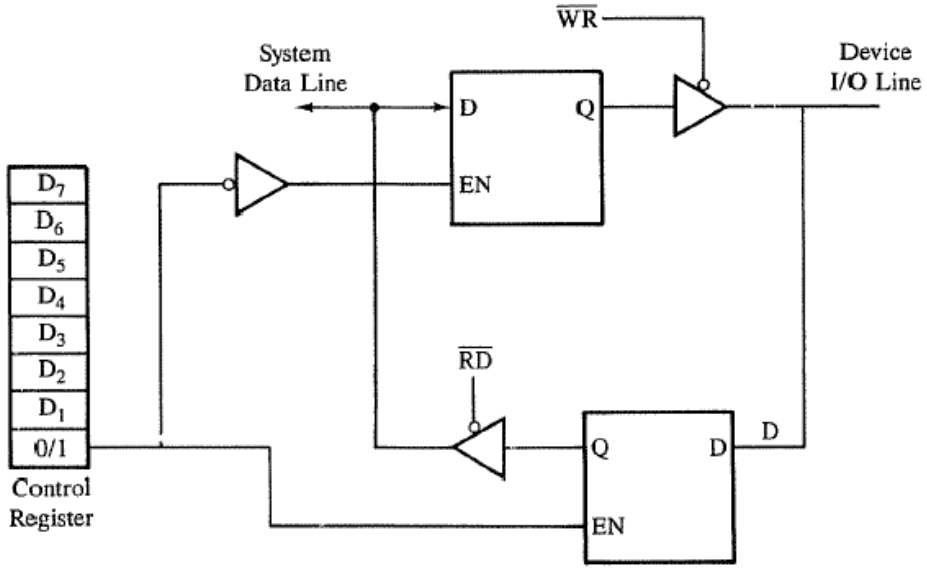


FIGURE 13.3
Programmable I/O Ports

Programmable device with Handshake Signals:-

The CPU and peripherals at different speeds, therefore signals are exchanged prior to data transfer between the fast response CPU and slow response of peripherals, such as printers and keyboard. These signal called handshake signals, these signals generally provide by programmable devices.

The steps in data input from a peripheral such as keyboard

1. A peripheral send data byte in the port and informs the interfacing device by sending handshake signal STB (strobe).
2. The interfacing device informs the peripheral that the input port is full - do not send the next byte until this has been read by sending handshake signal IBF (input buffer full).
3. The interfacing device informs the CPU that there is a byte available to read by sending interrupt signal (INTR).
4. The CPU reads the byte by sending control signal RD.

The steps in data output to a peripheral such as printer

1. The CPU writes a data byte into the port of programmable interfacing device by sending control signal WR.
2. Programmable interfacing device interrupt the CPU until the peripheral receive the byte by sending interrupt signal (INTR).
3. The interfacing device informs the peripheral that the byte is on the way by sending handshake signal OBF (output buffer full).
4. When the peripheral receive the byte it acknowledge the interfacing device by sending acknowledge signal ACK to interfacing device.
5. The interfacing device interrupts the CPU to ask for the next byte by sending interrupt signal (INTR).

The figure below explains the two processes (input and output).