Exercise-5: AT89C51ED2 kit

- Kit manual print out
- Flip software

PROGRAM DOWNLOADING (DOWNLOADING .HEX FILE TO THE BOAD) USING FLIP Software:

1. Set the slide switch **SW2** to PROG position on the board and press the reset switch **SW1**.
2. Open the Atmel Flip 2.4.2 tool by double clicking on "**Atmel Flip"** icon.
3. Goto "**Device**" option -> "**Select**", select the specific device "**AT89C51ED2**" & press **OK**.
4. Goto "**File**" -> "**Load HEX File**", navigate to desired **.HEX** file of the project.
5. Goto "**Settings**" option -> "**Communication**" -> "**RS232**", a window will open, and make sure that no other application is using COM port. Click on **COM** and select the proper COM port (Eg:COM1), set the baud rate to 115200 and click on **Connect**.
6. In "**Operations Flow**" region, check the options "**Erase**", "**Blank Check**", "**Program**" & "**Verify**".
7. In right most side of the window select the option **BLJB**, set the settings **BSB**, **EB**, **SBV** as **00**, **FF** and **FC** respectively and select the option "**Level 0**" in Device SSB region.

After selecting all the above click on **RUN** in field "**Operations Flow"** and wait until get the status as **finished** on the status bar. If an error occurs during programming then press **SW2** (reset) button and reprogram the device. After programming set the slide switch **SW1** towards RUN position on the board and Press **SW2** (reset) button to execute the program. If the communication error occurs during the programming then close the Atmel Flip 2.4.2 remove the RS232 cable and reinsert the cable and once again program the device. After programming successfully close the Atmel Flip window.

Caution: Do not reset the device during Flash Programming.

- Serial to USB converter cable
- RS232C Serial Port for Programming and serial communication
- Programming – Chip / ISP / IAP
- Hands On kit:
  - Shift Register implementation on At89c51ED2 kit

8b Shift Register

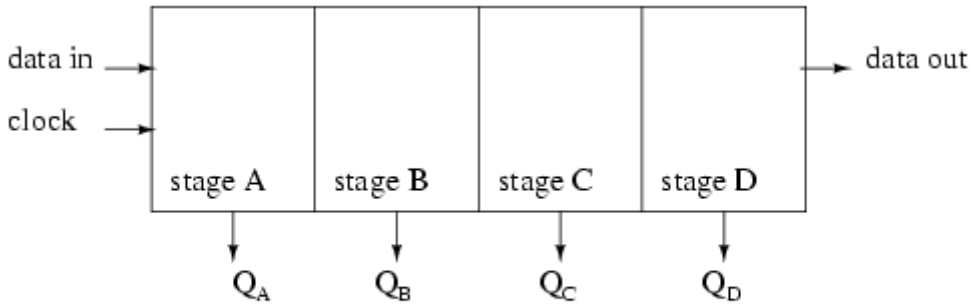
Serial-in, parallel-out shift register

A serial-in/parallel-out shift register is similar to the serial-in/ serial-out shift register in that it shifts data into internal storage elements and shifts data out at the serial-out, data-out, pin.

It is different in that it makes all the internal stages available as outputs.

Therefore, a serial-in/parallel-out shift register converts data from serial format to parallel format.

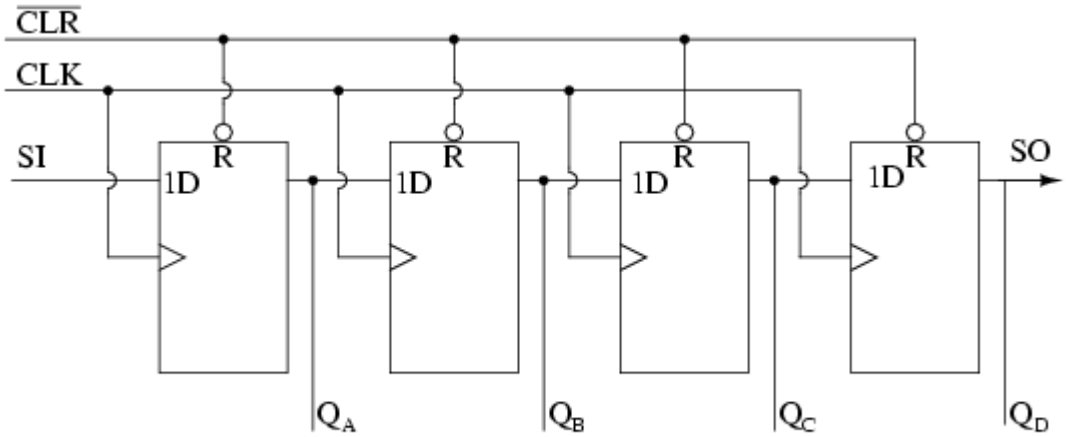
If four data bits are shifted in by four clock pulses via a single wire at data-in, below, the data becomes available simultaneously on the four Outputs Q_A to Q_D after the fourth clock pulse.



Serial-in, parallel-out shift register with 4-stages

The practical application of the serial-in/parallel-out shift register is to convert data from serial format on a single wire to parallel format on multiple wires.

Perhaps, we will illuminate four LEDs (Light Emitting Diodes) with the four outputs (**Q_A Q_B Q_C Q_D**).



Serial-in/ Parallel out shift register details

The above details of the serial-in/parallel-out shift register are fairly simple.

It looks like a serial-in/ serial-out shift register with taps added to each stage output.

Serial data shifts in at **SI** (Serial Input).

After a number of clocks equal to the number of stages, the first data bit in appears at SO (Q_D) in the above figure.

In general, there is no SO pin. The last stage (Q_D above) serves as SO and is cascaded to the next package if it exists.

If a serial-in/parallel-out shift register is so similar to a serial-in/ serial-out shift register, why do manufacturers bother to offer both types?

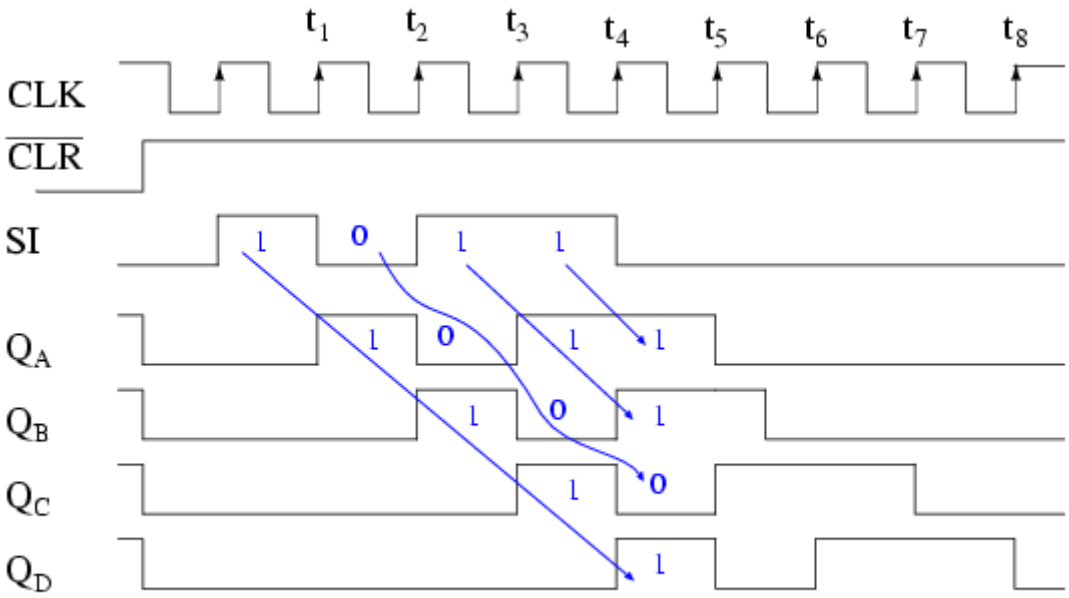
Why not just offer the serial-in/parallel-out shift register?

They actually only offer the serial-in/parallel-out shift register, as long as it has no more than 8-bits.

Note that serial-in/ serial-out shift registers come in bigger than 8-bit lengths of 18 to 64-bits.

It is not practical to offer a 64-bit serial-in/parallel-out shift register requiring that many output pins.

See waveforms below for above shift register.



Serial-in/ parallel-out shift register waveforms

The shift register has been cleared prior to any data by **CLR'**, an active low signal, which clears all type D Flip-Flops within the shift register.

Note the serial data **1011** pattern presented at the **SI** input.

This data is synchronized with the clock **CLK**.

This would be the case if it is being shifted in from something like another shift register, for example, a parallel-in/ serial-out shift register (not shown here).

On the first clock at **t1**, the data **1** at **SI** is shifted from **D** to **Q** of the first shift register stage.

After **t2** this first data bit is at $\mathbf{Q_B}$. After **t3** it is at $\mathbf{Q_C}$. After **t4** it is at $\mathbf{Q_D}$.

Four clock pulses have shifted the first data bit all the way to the last stage $\mathbf{Q_D}$.

The second data bit a **0** is at $\mathbf{Q_C}$ after the 4th clock.

The third data bit a **1** is at $\mathbf{Q_B}$.

The fourth data bit another **1** is at $\mathbf{Q_A}$.

Thus, the serial data input pattern **1011** is contained in ($\mathbf{Q_D}$ $\mathbf{Q_C}$ $\mathbf{Q_B}$ $\mathbf{Q_A}$).

It is now available on the four outputs.

It will available on the four outputs from just after clock $\mathbf{t_4}$ to just before $\mathbf{t_5}$.

This parallel data must be used or stored between these two times, or it will be lost due to shifting out the Q_D stage on following clocks $\mathbf{t_5}$ to $\mathbf{t_8}$ as shown above.

Serial-in/ parallel-out devices

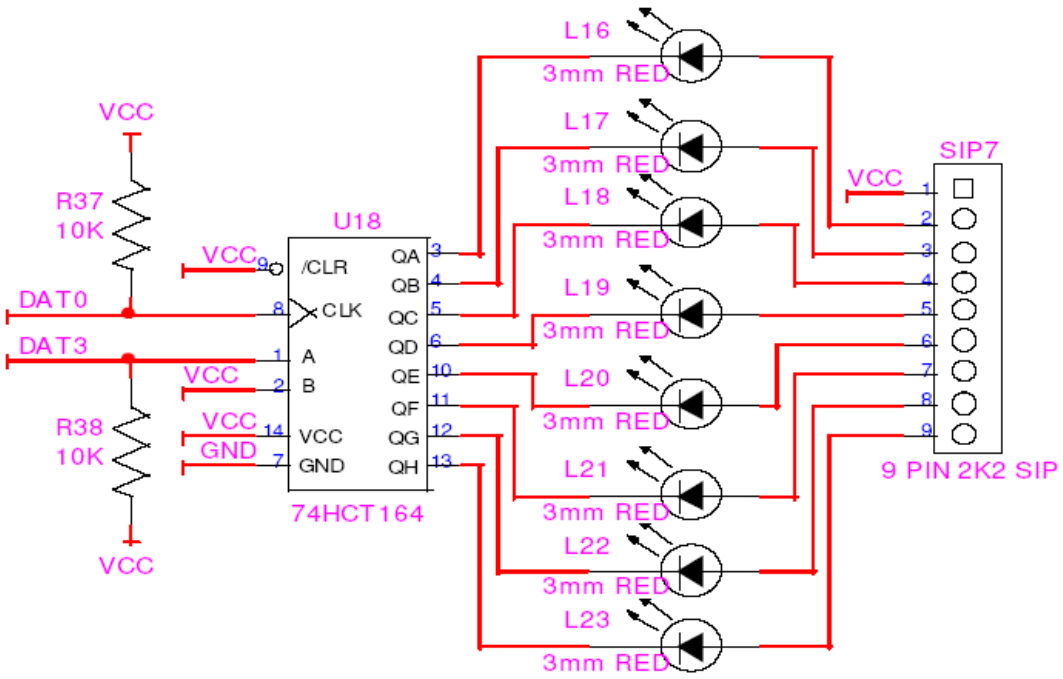
ALS AT89C51ED2 Kit

The Flash Programmable Embedded Controller Evaluation Board consists of, 8 LEDs interfaced using serial shift register.

Eight numbers of LED's (L16 to L23) interfaced through serial shift register

LEDs:

Interface Diagram:



Shift Register with LED interface

The LEDs L16 to L23 are configured as active low type, i.e. they turn ON to indicate a logical '0' status. The cathode pins of the LEDs are connected to the 8 output lines of 8-bit shift register 74HCT164 (**U18**) as given in the following table and anodes are connected to +5V VCC through 2K2 resistors **SIP7**. Port line P1.0 (CLK) is used as clock input to the 8-bit shift register **U18** and P0.3 (A) is used as data line. Toggling the 1.0 from '1' to '0' and back to `1' will transfer the data at Port line P0.3 to the first Q o/p of the shift register . Subsequent clocks will transfer the new data at the data i/p to the first o/p while the first o/p is shifted to the second and so on.

Connection Details:

LED No (Cathode)	8-Bit Shift Register (U5) Pin No.
L16	Pin-3
L17	Pin-4
L18	Pin-5
L19	Pin-6
L20	Pin-10
L21	Pin-11
L22	Pin-12
L23	Pin-13

//Pseudo code

```
//P1_0 is the clock input
//P1_3 is the data input
//do infinitely
        //clear all leds
                //set data input as 1 and apply 8 clock pulse
        //set all leds
                //set data input as 0 and apply 8 clock pulse
```

//sample code

```
sbit CLKL = P1^0;          //Clock Line (>CLK)of 8-bit shift reg U5 (Serial In Parralel Out)
sbit LED = P1^3;           //Data Line of the 8-bit shift reg U5
unsigned int j;

void main()
{
        //clear all leds
        CLKL = 0;          //clear clock input intially
        LED = 1;           //set data input as 1
        for(j=0;j<8;j++) //apply 8 clock pulses
        {
                CLKL = 1;
                CLKL  =0;
        }

        //set all leds
        CLKL = 0; //clear clock input initially
        LED = 0; //set data input as 0
        //apply 8 clock pulses
        for(j=0;j<8;j++)
        {
                CLKL = 1;
                CLKL  =0;
        }

}
```