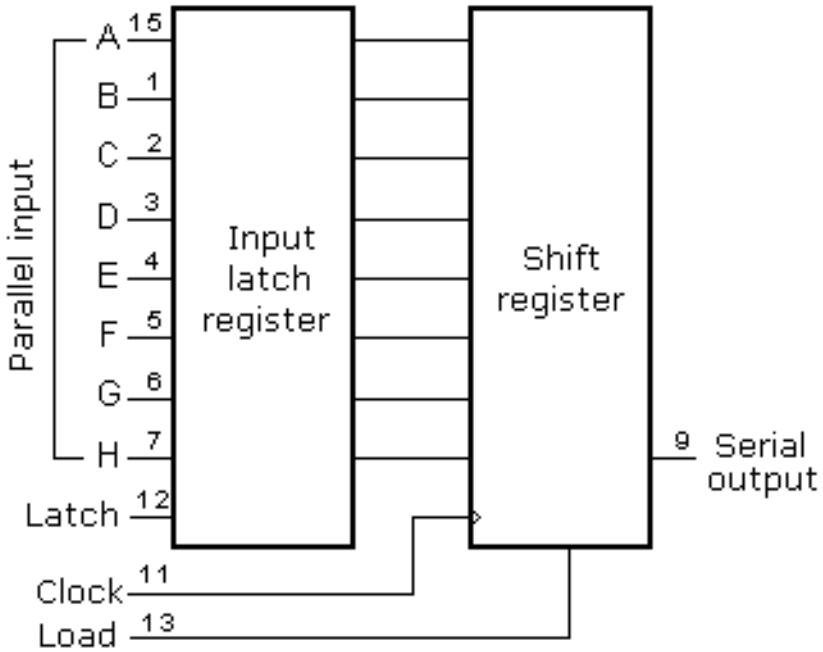
Shift registers

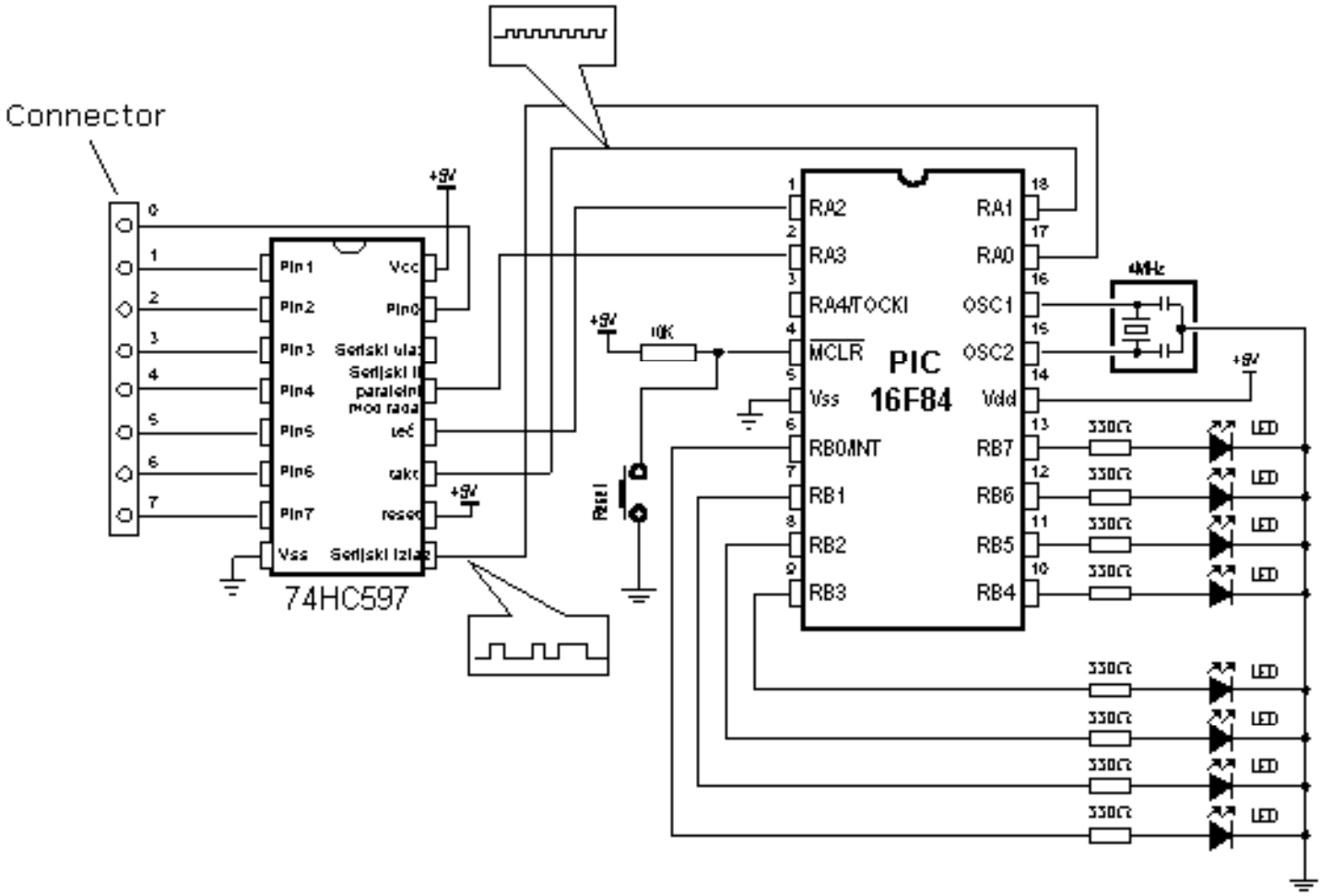
There are two types of shift registers: **input and output**. **Input shift registers** receive data in parallel, through 8 lines and then send it serially through two lines to a microcontroller. **Output shift registers** work in the opposite direction; they receive serial data and on a "latch" line signal, they turn it into parallel data. Shift registers are generally used to increase the number of input-output lines of a microcontroller. They are not so much in use any more though, because most modern microcontrollers have a large number of input/output lines. However, their use with microcontrollers such as PIC16F84 can be justified.

Input shift register 74HC597

Input shift registers transform parallel data into serial data and transfer it to a microcontroller. Their working is quite simple. There are four lines for the transfer of data: **clock, latch, load and data**. Data is first read from the input pins by an internal register through a 'latch' signal. Then, with a 'load' signal, data is transferred from the input latch register to the shift register, and from there it is serially transferred to a microcontroller via 'data' and 'clock' lines.



An outline of the connection of the shift register 74HC597 to a micro, is shown below.



How to connect an input shift register to a microcontroller

In order to simplify the main program, a macro can be used for the input shift register. Macro HC597 has two arguments:

HC597 macro Var, Var1

Var variable where data from input pins is transferred
Var1 loop counter

Example: HC597 data, counter

Data from the input pins of the shift register is stored in data variable. Timer/counter variable is used as a loop counter.

Macro listing:

Makro: HC597.INC

```
HC597  macro   Var,Varl

       Local  Loop          ; local label

       movlw  .8            ; transfer eight bits
       movwf  Varl          ; counter initialization

       bsf    Latch         ; receive status from pins at input latch
       nop
       bcf    Latch

       bcf    Load
       nop
       bsf    Load

Loop   rlf    Var,f         ; Rotate 'Var' one space to the left

       btfss  Data          ; Is Data line = '1' ?
       bcf    Var,0         ; If not, set erase bit '0' at Var variable
       btfsc  Data          ; Is Dataline = '0'?
       bsf    Var,0         ; If not set bit '0'

       bsf    Clock         ; make one clock
       nop
       bcf    Clock

       decfsz Varl,f        ; are 8 bits received?
       goto   Loop          ; if not, repeat

       endm
```

Example of how to use the HC597 macro is given in the following program. Program receives data from a parallel input of the shift register and moves it serially into the RX variable of the microcontroller. LEDs connected to port B will indicate the result of the data input.

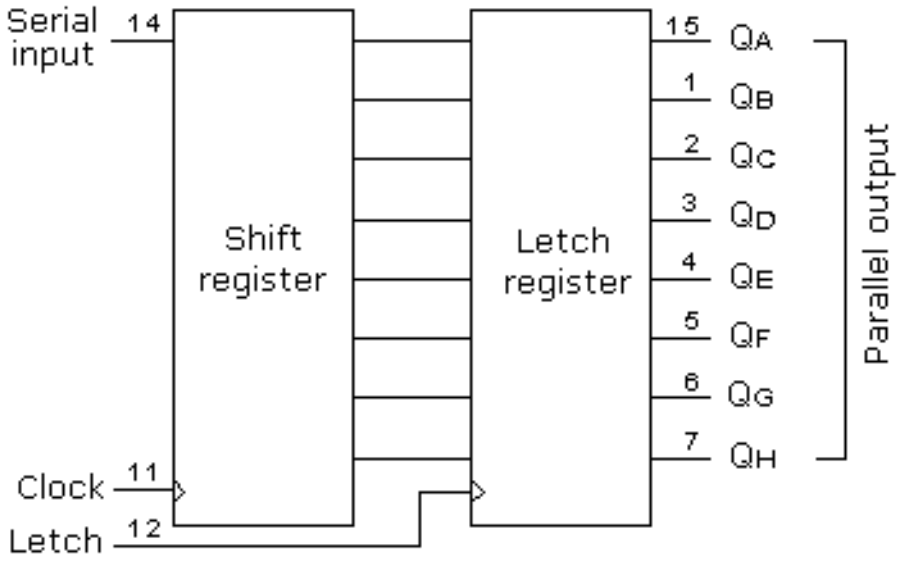
Program: HC597.INC

```
;***** Declaration and configuration of microcontroller *****

       PROCESSOR 16f84
       #include "p16f84.inc"

       __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

;***** Declaring the variables *****

       Cblock  0x0C     ; beginning of RAM
       RX
```

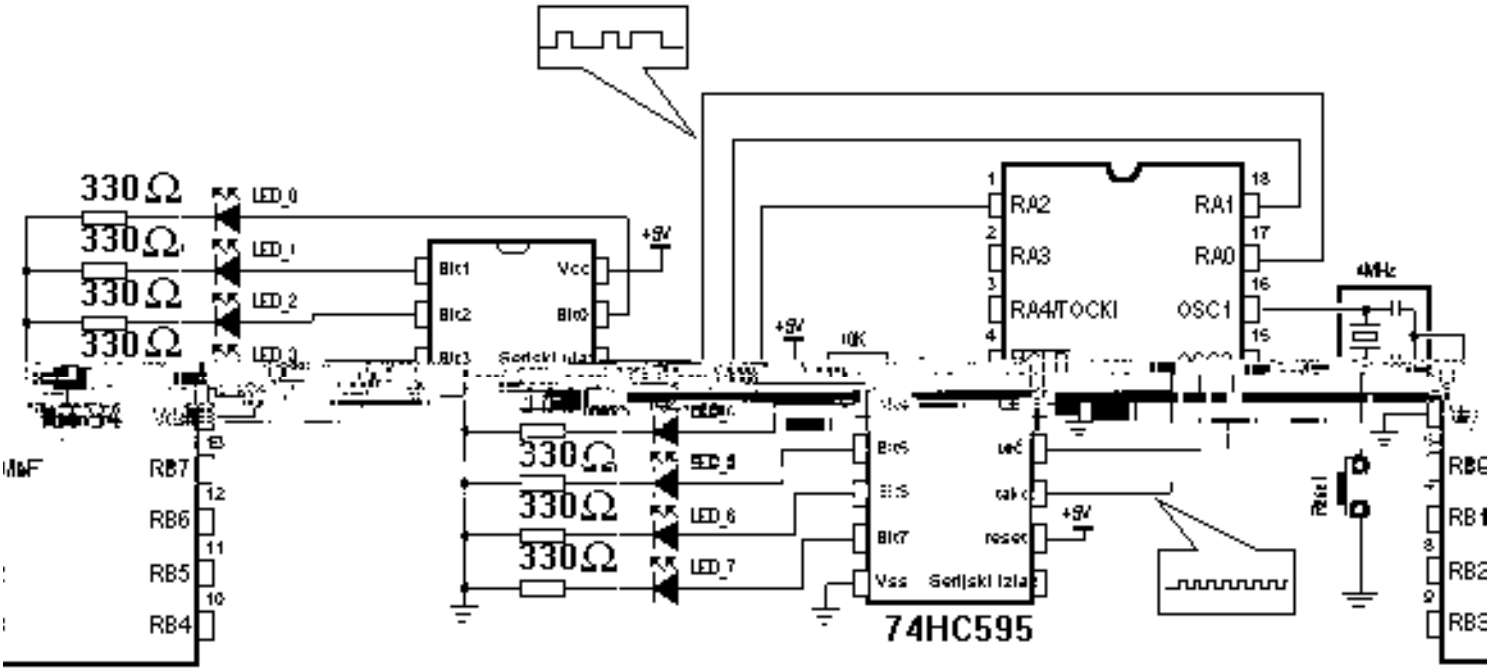
```
        Cblock  0x0C    ; beginning of RAM
        RX
        CountSPI
        endc

;***** Declaring the hardware *****

        #define Data PORTA,0        ; can be any other I/O pin
        #define Clock PORTA,1
        #define Latch PORTA,2
        #define Load PORTA,3

;***** Program memory structure *****

        ORG     0x00                ; reset vector
        goto    Main

        ORG     0x04                ; Interrupt vector
        goto    Main                ; no interrupt routine

        #include "bank.inc"         ; assistant files
        #include "hc597.inc"

Main                                ; beginning of a program
        BANK1
        movlw   b'00010001'         ; port A initialization
        movwf   TRISA               ; TRISA <- 0x11
        clrf    TRISB               ; pins of port B
        BANK0

        clrf    PORTA               ; PORTA <- 0x00
        bsf     Load                ; Enable SHIFT register

Loop    HC597   RX, CountSPI        ; Status of input pins of SHIFT register
        movf    RX,W                ; Are found in variable RX
        movwf   PORTB               ; Set the contents of RX register to
                                    ; port B

        goto    Loop                ; Repeat the loop
        End                         ; End of program
```

Output shift register

Output shift registers transform serial data into parallel data. On every rising edge of the clock, the shift register reads the value from data line, stores it in temporary register, and then repeats this cycle 8 times. On a signal from 'latch' line, data is copied from the shift register to input register, thus data is transformed from serial into parallel data.



An outline of the 74HC595 shift register connections is shown on the diagram below:



Example: HC595 Data, counter

The data we want to transfer is stored in data variable, and counter variable is used as a loop counter.

Makro: HC595.INC

```
HC595  macro  Var,Varl

       Local  Loop          ; local label
       movlw  .8            ; transfer eight bits
       movwf  Varl          ; counter initialization

Loop   rlf    Var,f         ; Rotate 'Var' one space to the left

       btfss  STATUS,C      ; Is carry = '1' ?
       bcf    Data          ; If not, set Data line to '0'
       btfsc  STATUS,C      ; Is carry = '0' ?
       bsf    Data          ; If not, set Data line to '1'

       bsf    Clock         ; Make one clock
       nop
       bcf    Clock

       decfsz Varl,f        ; Are eight bits sent?
       goto   Loop          ; If not, repeat

       bsf    Latch         ; If all 8 bits have been sent, move the
       nop                  ; contents from SHIFT register to output latch
       bcf    Latch

       endm
```

An example of how to use the HC595 macro is given in the following program. Data from variable TX is serially transferred to shift register. LEDs connected to the parallel output of the shift register will indicate the state of the lines. In this example value 0xCB (1100 1011) is sent so that the eighth, seventh, fourth, second and first LEDs are illuminated.

Program: HC595.ASM

```
;***** Microcontroller configuration and declaration *****

       PROCESSOR 16f84
       #include "p16f84.inc"

       __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

;***** Declaring the variables *****

       Cblock 0x0C                  ; Beginning of RAM
       TX                           ; Belongs to function "HC595"
       CountSPI
       endc

;***** Declaring the hardware *****

       #define Data PORTA,0
       #define Clock PORTA,1
       #define Latch PORTA,2

;***** Structure of program memory *****

       ORG    0x00                  ; Reset vector
       goto   Main
       ORG    0x04                  ; Interrupt vector
       goto   Main                  ; There is no interrupt routine

       #include "bank.inc"          ; Assistant files
       #include "hc595.inc"
Main                                ; Beginning of the program
       BANK1
       movlw  b'00011000'           ; Port A initialization
       movwf  TRISA                 ; TRISA <- 0x18
       BANK0
       clrf   PORTA                 ; PORTA <- 0x00
       movlw  0xcb                  ; Fill the TX buffer
       movwf  TX                    ; TX <- '11001011'
       HC595  TX, CountSPI
Loop   goto   Loop                  ; Stay here
       End                          ; End of program
```