PIC Tutorial Five - Infrared Communication

To complete all of these tutorials you will require two Main Boards, two IR Boards, the LCD Board, the Switch Board, and the LED Board, as written the first two tutorials use the LCD Board and Switch Board on PortA and the IR Boards on PortB - although these could easily be swapped over, as the IR Board doesn't use either of the two 'difficult' pins for PortA, pins 4 and 5. The third tutorial uses the IR Board on PortA and the LED Board on PortB (as we require all 8 pins to be outputs). Download zipped tutorial files.

IR transmission has limitations, the most important one (for our purposes) being that the receiver doesn't give out the same width pulses that we transmit, so we can't just use a normal, RS232 type, serial data stream, where we simply sample the data at fixed times - the length of the received data varies with the number of ones sent - making receiving it accurately very difficult. Various different schemes are used by the manufacturers of IR remote controls, and some are much more complicated than others.

I've chosen to use the Sony SIRC (Sony Infra Red Control) remote control system, many of you may already have a suitable Sony remote at home you can use, and it's reasonably easy to understand and implement. Basically it uses a pulse width system, with a start bit of 2.4mS, followed by 12 data bits, where a '1' is 1.2mS wide, and a '0' is 0.6mS wide, the bits are all separated by gaps of 0.6mS. The data itself consists of a 7 bit 'command' code, and a 5 bit 'device' code - where a command is Channel 1, Volume Up etc. and a device is TV, VCR etc. This is how the same remote system can be used for different appliances, the same command for 'Power On' is usually used by all devices, but by transmitting a device ID only a TV will respond to 'TV Power On' command.

Start	Command Code							Device Code				
S	D0	D1	D2	D3	D4	D5	D6	C0	C1	C2	C3	C4
2.4mS	1.2 or 0.6mS							1.2 or 0.6mS				

The table to the right shows the data format, after the Start bit the command code is send, lowest bit first, then the device code, again lowest bit first. The entire series is sent repeatedly while the button is held down, every 45mS. In order to decode the transmissions we need to measure the width of the pulses, first looking for the long 'start' pulse, then measuring the next 12 pulses and deciding if they are 1's or 0's. To do this I'm using a simple software 8 bit counter, with NOP's in the loop to make sure we don't overflow the counter. After measuring one pulse we then test it to see if it's a valid pulse, this routine provides four possible responses 'Start Pulse', 'One', 'Zero', or 'Error', we initially loop until we get a 'Start Pulse' reply, then read the next 12 bits - if the reply to any of these 12 is other than 'One' or 'Zero' we abort the read and go back to waiting for a 'Start Pulse'.

Device ID's

TV	1
VTR1	2
Text	3
Widescreen	4
MDP	6
VTR2	7
VTR3	11
Effect	12
Audio	16
Pro-Logic	18
DVD	26

The device codes used specify the particular device, but with a few exceptions!, while a TV uses device code 1, some of the Teletext buttons use code 3, as do the Fastext coloured keys - where a separate Widescreen button is fitted, this uses code 4. The table to the left shows some of the Device ID codes I found on a sample of Sony remotes. Five bits gives a possible 32 different device ID's, and some devices respond to more than one device ID, for example some of the current Sony VCR's have the Play button in a 'cursor' type of design, surrounded by 'Stop', 'Pause', 'Rewind', and 'Fast Forward' - the ones I tested actually send a DVD ID code when these keys are pressed (along with a different command ID to that used normally used for 'Play' etc.). However, they still respond to an older Sony remote which sends the VTR3 device ID, which despite being labelled VTR3 on TV remotes seems to be the normal standard Sony VCR device ID. It's quite common for Sony remotes to use more than one device ID, a Surround Sound Amplifier Remote I tried used four different device ID's.

If you don't have a Sony remote you can use, I've also built a transmitter, using the second Main Board, second IR Board, and the Switch Board, the four buttons allow you to send four different command codes - I've chosen TV as the device, and Volume Up, Volume Down, Program Up, and Program Down as my four commands, I've confirmed this works on various Sony TV's. Transmitting the SIRC code is quite simple to do, I generate the 38KHz modulation directly in software, and to reduce current consumption don't use a 50/50 on/off ratio - by using a longer off than on time we still get the 38KHz, but with a reduced power requirement.

UPDATED I've recently discovered that Sony DVD players DON'T use the standard 12 bit SIRC's system, it's extended to comprise 20 bits instead. It still has the same 5 bit device code and 7 bit command code, but it's followed by an extra 8 bit code at the end. In the ones I've tested these 8 bits were always the same, hexadecimal 0x49. It's simple to add this to the transmitter, just add an extra section after the device code section 'Ser_Loop2' that sends 8 bits with the value 0x49. Apparently there's also a third variant of SIRC's that uses 15 bits, a 7 bit command code, and an 8 bit device code. So, all together, three versions, 12 bit, 15 bit, and 20 bit, although 12 bit seems by far the most common, DVD players seem to use 20 bit, and I've yet to see a 15 bit remote.

Another interesting Sony point is that some remotes can be configured to act as 'service remotes', this changes one of the buttons to become a 'test' button, pressing it once displays 'T' on the screen, pressing it twice displays 'TT' and enters test mode - pressing 'Menu' at this point displays the service menu. In order to make yourself a 'service remote' you just need to send the Device ID '1' and the command '127'.

Tutorial 5.1 - requires one Main Board (with LED set to RB7), one IR Board and LCD Board.

This program uses the LCD module to give a decimal display of the values of the Device and Command bytes transmitted by a Sony SIRC remote control, it can be easily altered to operate port pins to control external devices, as an example the main board LED is turned on by pressing button 2, turned off by pressing button 3, and toggled on and off by pressing button 1 (all on a TV remote, you can change the device ID for a different remote if you need to). As it stands it's very useful for displaying the data transmitted by each button on your Sony remote control - the **Device ID's** table above was obtained using this design.

```
;Tutorial 5_1
;Read SIRC IR with LCD display
;Nigel Goodwin 2002

        LIST    p=16F628                ;tell assembler what chip we are using
        include "P16F628.inc"           ;include the defaults for the chip
        ERRORLEVEL      0,      -302    ;suppress bank selection messages
        __config 0x3D18                 ;sets the configuration settings (oscillator type etc.)




                cblock  0x20                    ;start of general purpose registers
                        count                   ;used in looping routines
                        count1                  ;used in delay routine
                        counta                  ;used in delay routine
                        countb                  ;used in delay routine
                        LoX
                        Bit_Cntr
                        Cmd_Byte
                        Dev_Byte
                        Timer_H
                        Flags
                        Flags2
                        tmp1                    ;temporary storage
                        tmp2
                        tmp3
                        lastdev
                        lastkey

                        NumL                    ;Binary inputs for decimal convert routine
                        NumH

                        TenK                    ;Decimal outputs from convert routine
                        Thou
                        Hund
                        Tens
                        Ones

                        templcd                 ;temp store for 4 bit mode
                        templcd2
                endc
```

```
LCD_PORT        Equ     PORTA
LCD_TRIS        Equ     TRISA
LCD_RS          Equ     0x04                    ;LCD handshake lines
LCD_RW          Equ     0x06
LCD_E           Equ     0x07

IR_PORT         Equ     PORTB
IR_TRIS         Equ     TRISB
IR_In           Equ     0x02                    ;input assignment for IR data

OUT_PORT        Equ     PORTB
LED             Equ     0x07

ErrFlag         Equ     0x00
StartFlag       Equ     0x01                    ;flags used for received bit
One             Equ     0x02
Zero            Equ     0x03

New             Equ     0x07                    ;flag used to show key released

TV_ID           Equ     0x01                    ;TV device ID

But1            Equ     0x00                    ;numeric button ID's
But2            Equ     0x01
But3            Equ     0x02
But4            Equ     0x03
But5            Equ     0x04
But6            Equ     0x05
But7            Equ     0x06
But8            Equ     0x07
But9            Equ     0x08

                org     0x0000
                goto    Start

                org     0x0004
                retfie

;TABLES - moved to start of page to avoid paging problems,
;a table must not cross a 256 byte boundary.
HEX_Table       addwf   PCL       , f
                retlw   0x30
                retlw   0x31
                retlw   0x32
                retlw   0x33
                retlw   0x34
                retlw   0x35
                retlw   0x36
                retlw   0x37
                retlw   0x38
                retlw   0x39
                retlw   0x41
                retlw   0x42
                retlw   0x43
                retlw   0x44
                retlw   0x45
                retlw   0x46

Xtext           addwf   PCL, f
                retlw   'D'
                retlw   'e'
                retlw   'v'
                retlw   'i'
                retlw   'c'
                retlw   'e'
                retlw   ' '
                retlw   ' '
                retlw   ' '
                retlw   'C'
                retlw   'o'
                retlw   'm'
                retlw   'm'
                retlw   'a'
                retlw   'n'
                retlw   'd'
                retlw   0x00

;end of tables

Start           movlw   0x07
                movwf   CMCON                   ;turn comparators off (make it like a 16F84)

Initialise      clrf    count
                clrf    PORTA
                clrf    PORTB
                clrf    Flags
                clrf    Dev_Byte
                clrf    Cmd_Byte



SetPorts        bsf     STATUS,         RP0     ;select bank 1
                movlw   0x00                    ;make all LCD pins outputs
                movwf   LCD_TRIS
                movlw   b'01111111'             ;make all IR port pins inputs (except RB7)
                movwf   IR_TRIS
                bcf     STATUS,         RP0     ;select bank 0

                call    LCD_Init                ;setup LCD module
                call    Delay255                ;let IR receiver settle down

Main
                call    LCD_Line1               ;set to first line
                call    String1                 ;display IR title string

                call    ReadIR                  ;read IR signal
                movlw   d'2'
                call    LCD_Line2W              ;set cursor position
                clrf    NumH
                movf    Dev_Byte,       w       ;convert device byte
                movwf   NumL
                call    Convert
                movf    Tens,   w
                call    LCD_CharD
                movf    Ones,   w
                call    LCD_CharD

                movlw   d'11'
                call    LCD_Line2W              ;set cursor position
                clrf    NumH
                movf    Cmd_Byte,       w       ;convert data byte
                movwf   NumL
                call    Convert
                movf    Hund,   w
                call    LCD_CharD
                movf    Tens,   w
                call    LCD_CharD
                movf    Ones,   w
                call    LCD_CharD

                call    ProcKeys                ;do something with commands received

                goto    Main                    ;loop for ever

ProcKeys
                btfss   Flags2, New
                retlw   0x00                    ;return if not new keypress
                movlw   TV_ID                   ;check for TV ID code
                subwf   Dev_Byte,       w
                btfss   STATUS    , Z
                retlw   0x00                    ;return if not correct code

                movlw   But1                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key1                    ;try next key if not correct code

                movf    OUT_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED             ;and test LED bit for toggling
                bsf     OUT_PORT,       LED     ;turn on LED
                btfsc   tmp3,   LED
                bcf     OUT_PORT,       LED     ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                retlw   0x00

Key1            movlw   But2                    ;test for button 2
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key2                    ;try next key if not correct code
                                                ;this time just turn it on
                bsf     OUT_PORT,       LED     ;turn on LED
                bcf     Flags2, New             ;and cancel new flag
                retlw   0x00

Key2            movlw   But3                    ;test for button 3
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                retlw   0x00                    ;return if not correct code
                                                ;this time just turn it off
                bcf     OUT_PORT,       LED     ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                retlw   0x00

String1         clrf    count                   ;set counter register to zero
Mess1           movf    count, w                ;put counter value in W
                call    Xtext                   ;get a character from the text table
                xorlw   0x00                    ;is it a zero?
                btfsc   STATUS, Z
                retlw   0x00                    ;return when finished
                call    LCD_Char
                incf    count, f
                goto    Mess1

;IR routines

ReadIR          call    Read_Pulse
                btfss   Flags,  StartFlag
                goto    ReadIR                  ;wait for start pulse (2.4mS)

Get_Data        movlw   0x07                    ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Cmd_Byte
Next_RcvBit2    call    Read_Pulse
                btfsc   Flags,  StartFlag       ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,  ErrFlag         ;abort if error
                goto    ReadIR

                bcf     STATUS    , C
                btfss   Flags,  Zero
                bsf     STATUS    , C
                rrf     Cmd_Byte  , f
                decfsz  Bit_Cntr  , f
                goto    Next_RcvBit2
                rrf     Cmd_Byte  , f           ;correct bit alignment for 7 bits

Get_Cmd         movlw   0x05                    ;set up to read 5 bits
                movwf   Bit_Cntr
                clrf    Dev_Byte
Next_RcvBit     call    Read_Pulse
                btfsc   Flags,  StartFlag       ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,  ErrFlag         ;abort if error
                goto    ReadIR

                bcf     STATUS    , C
                btfss   Flags,  Zero
                bsf     STATUS    , C
                rrf     Dev_Byte  , f
                decfsz  Bit_Cntr  , f
                goto    Next_RcvBit

                rrf     Dev_Byte  , f           ;correct bit alignment for 5 bits
                rrf     Dev_Byte  , f
                rrf     Dev_Byte  , f

                retlw   0x00

;end of ReadIR


;read pulse width, return flag for StartFlag, One, Zero, or ErrFlag
;output from IR receiver is normally high, and goes low when signal received

Read_Pulse      clrf    LoX
                btfss   IR_PORT,        IR_In   ;wait until high
                goto    $-1
                clrf    tmp1
                movlw   0xC0                    ;delay to decide new keypress
                movwf   tmp2                    ;for keys that need to toggle

Still_High      btfss   IR_PORT,        IR_In   ;and wait until goes low
                goto    Next
                incfsz  tmp1,f
                goto    Still_High
                incfsz  tmp2,f
                goto    Still_High
                bsf     Flags2, New             ;set New flag if no button pressed
                goto    Still_High

Next            nop
                nop
                nop
                nop
                nop                             ;waste time to scale pulse
                nop                             ;width to 8 bits
                nop
                nop
                nop
                nop
                nop
                nop
                incf    LoX,    f
                btfss   IR_PORT,        IR_In
                goto    Next                    ;loop until input high again

; test if Zero, One, or Start (or error)

Chk_Pulse       clrf    Flags

TryError        movf    LoX,    w               ; check if pulse too small
                addlw   d'255' - d'20'          ; if LoX <= 20
                btfsc   STATUS    , C
                goto    TryZero
                bsf     Flags,  ErrFlag         ; Error found, set flag
                retlw   0x00

TryZero         movf    LoX,    w               ; check if zero
                addlw   d'255' - d'60'          ; if LoX <= 60
                btfsc   STATUS    , C
                goto    TryOne
                bsf     Flags,  Zero            ; Zero found, set flag
                retlw   0x00

TryOne          movf    LoX,    w               ; check if one
                addlw   d'255' - d'112'         ; if LoX <= 112
                btfsc   STATUS    , C
                goto    TryStart
                bsf     Flags,  One             ; One found, set flag
                retlw   0x00

TryStart        movf    LoX,    w               ; check if start
                addlw   d'255' - d'180'         ; if LoX <= 180
                btfsc   STATUS    , C
                goto    NoMatch
                bsf     Flags,  StartFlag       ; Start pulse found
                retlw   0x00
NoMatch                                         ; pulse too long
                bsf     Flags,  ErrFlag         ; Error found, set flag
                retlw   0x00

;end of pulse measuring routines

;LCD routines

;Initialise LCD
LCD_Init        call    LCD_Busy                ;wait for LCD to settle

                movlw   0x20                    ;Set 4 bit mode
                call    LCD_Cmd

                movlw   0x28                    ;Set display shift
                call    LCD_Cmd

                movlw   0x06                    ;Set display character mode
                call    LCD_Cmd

                movlw   0x0c                    ;Set display on/off and cursor command
                call    LCD_Cmd                 ;Set cursor off

                call    LCD_Clr                 ;clear display

                retlw   0x00

; command set routine
LCD_Cmd         movwf   templcd
                swapf   templcd,        w       ;send upper nibble
                andlw   0x0f                    ;clear upper 4 bits of W
                movwf   LCD_PORT
                bcf     LCD_PORT, LCD_RS        ;RS line to 1
                call    Pulse_e                 ;Pulse the E line high

                movf    templcd,        w       ;send lower nibble
                andlw   0x0f                    ;clear upper 4 bits of W
                movwf   LCD_PORT
                bcf     LCD_PORT, LCD_RS        ;RS line to 1
                call    Pulse_e                 ;Pulse the E line high
                call    LCD_Busy
                retlw   0x00

LCD_CharD       addlw   0x30                    ;add 0x30 to convert to ASCII
LCD_Char        movwf   templcd
```

```
                swapf   templcd,        w       ;send upper nibble
                andlw   0x0f                    ;clear upper 4 bits of W
                movwf   LCD_PORT
                bsf     LCD_PORT, LCD_RS        ;RS line to 1
                call    Pulse_e                 ;Pulse the E line high

                movf    templcd,        w       ;send lower nibble
                andlw   0x0f                    ;clear upper 4 bits of W
                movwf   LCD_PORT
                bsf     LCD_PORT, LCD_RS        ;RS line to 1
                call    Pulse_e                 ;Pulse the E line high
                call    LCD_Busy
                retlw   0x00

LCD_Line1       movlw   0x80                    ;move to 1st row, first column
                call    LCD_Cmd
                retlw   0x00

LCD_Line2       movlw   0xc0                    ;move to 2nd row, first column
                call    LCD_Cmd
                retlw   0x00

LCD_Line1W      addlw   0x80                    ;move to 1st row, column W
                call    LCD_Cmd
                retlw   0x00

LCD_Line2W      addlw   0xc0                    ;move to 2nd row, column W
                call    LCD_Cmd
                retlw   0x00

LCD_CurOn       movlw   0x0d                    ;Set display on/off and cursor command
                call    LCD_Cmd
                retlw   0x00

LCD_CurOff      movlw   0x0c                    ;Set display on/off and cursor command
                call    LCD_Cmd
                retlw   0x00

LCD_Clr         movlw   0x01                    ;Clear display
                call    LCD_Cmd
                retlw   0x00

LCD_HEX         movwf   tmp1
                swapf   tmp1,   w
                andlw   0x0f
                call    HEX_Table
                call    LCD_Char
                movf    tmp1, w
                andlw   0x0f
                call    HEX_Table
                call    LCD_Char
                retlw   0x00

Pulse_e         bsf     LCD_PORT, LCD_E
                nop
                bcf     LCD_PORT, LCD_E
                retlw   0x00

LCD_Busy
                bsf     STATUS, RP0             ;set bank 1
                movlw   0x0f                    ;set Port for input
                movwf   LCD_TRIS
                bcf     STATUS, RP0             ;set bank 0
                bcf     LCD_PORT, LCD_RS        ;set LCD for command mode
                bsf     LCD_PORT, LCD_RW        ;setup to read busy flag
                bsf     LCD_PORT, LCD_E
                swapf   LCD_PORT, w             ;read upper nibble (busy flag)
                bcf     LCD_PORT, LCD_E
                movwf   templcd2
                bsf     LCD_PORT, LCD_E         ;dummy read of lower nibble
                bcf     LCD_PORT, LCD_E
                btfsc   templcd2, 7             ;check busy flag, high = busy
                goto    LCD_Busy                ;if busy check again
                bcf     LCD_PORT, LCD_RW
                bsf     STATUS, RP0             ;set bank 1
                movlw   0x00                    ;set Port for output
                movwf   LCD_TRIS
                bcf     STATUS, RP0             ;set bank 0
                return

;end of LCD routines


;Delay routines

Delay255        movlw   0xff            ;delay 255 mS
                goto    d0
Delay100        movlw   d'100'          ;delay 100mS
                goto    d0
Delay50         movlw   d'50'           ;delay 50mS
                goto    d0
Delay20         movlw   d'20'           ;delay 20mS
                goto    d0
Delay5          movlw   0x05            ;delay 5.000 ms (4 MHz clock)
d0              movwf   count1
d1              movlw   0xC7
                movwf   counta
                movlw   0x01
                movwf   countb
Delay_0         decfsz  counta, f
                goto    $+2
                decfsz  countb, f
                goto    Delay_0

                decfsz  count1  ,f
                goto    d1
                retlw   0x00

;end of Delay routines

;This routine downloaded from http://www.piclist.com
Convert:                        ; Takes number in NumH:NumL
                                ; Returns decimal in
                                ; TenK:Thou:Hund:Tens:Ones
        swapf   NumH, w
        iorlw   B'11110000'
        movwf   Thou
        addwf   Thou,f
        addlw   0XE2
        movwf   Hund
        addlw   0X32
        movwf   Ones

        movf    NumH,w
        andlw   0X0F
        addwf   Hund,f
        addwf   Hund,f
        addwf   Ones,f
        addlw   0XE9
        movwf   Tens
        addwf   Tens,f
        addwf   Tens,f

        swapf   NumL,w
        andlw   0X0F
        addwf   Tens,f
        addwf   Ones,f
        rlf     Tens,f
        rlf     Ones,f
        comf    Ones,f
        rlf     Ones,f

        movf    NumL,w
        andlw   0X0F
        addwf   Ones,f
        rlf     Thou,f

        movlw   0X07
        movwf   TenK

                    ; At this point, the original number is
                    ; equal to
                    ; TenK*10000+Thou*1000+Hund*100+Tens*10+Ones
                    ; if those entities are regarded as two's
                    ; complement binary.  To be precise, all of
                    ; them are negative except TenK.  Now the number
                    ; needs to be normalized, but this can all be
                    ; done with simple byte arithmetic.

        movlw   0X0A                             ; Ten
Lb1:
        addwf   Ones,f
        decf    Tens,f
        btfss   3,0
        goto   Lb1
Lb2:
        addwf   Tens,f
        decf    Hund,f
        btfss   3,0
        goto   Lb2
Lb3:
        addwf   Hund,f
        decf    Thou,f
        btfss   3,0
        goto   Lb3
Lb4:
        addwf   Thou,f
        decf    TenK,f
        btfss   3,0
        goto   Lb4

        retlw   0x00


                end
```

Tutorial 5.2 - requires one Main Board, one IR Board and Switch Board.

This program implements a Sony SIRC IR transmitter, pressing one of the four buttons sends the corresponding code, you can alter the codes as you wish, for this example I chose Volume Up and Down, and Program Up and Down. In order to use this with the LED switching above, I would suggest setting the buttons to transmit '1', '2', '3' and '4', where '4' should have no effect on the LED - the codes are 0x00, 0x01, 0x02, 0x03 respectively (just to confuse us, the number keys start from zero, not from one).

```
;Tutorial 5.2 - Nigel Goodwin 2002
;Sony SIRC IR transmitter
        LIST    p=16F628                ;tell assembler what chip we are using
        include "P16F628.inc"           ;include the defaults for the chip
        __config 0x3D18                 ;sets the configuration settings (oscillator type etc.)

        cblock  0x20                    ;start of general purpose registers
                count1                  ;used in delay routine
                counta                  ;used in delay routine
                countb
                count
                Delay_Count
                Bit_Cntr
                Data_Byte
                Dev_Byte
                Rcv_Byte
                Pulse
        endc


IR_PORT Equ     PORTB
IR_TRIS Equ     TRISB
IR_Out  Equ     0x01
IR_In   Equ     0x02
Ser_Out Equ     0x01
Ser_In  Equ     0x02
SW1     Equ     7                       ;set constants for the switches
SW2     Equ     6
SW3     Equ     5
SW4     Equ     4

TV_ID           Equ     0x01                    ;TV device ID

But1            Equ     0x00                    ;numeric button ID's
But2            Equ     0x01
But3            Equ     0x02
But4            Equ     0x03
But5            Equ     0x04
But6            Equ     0x05
But7            Equ     0x06
But8            Equ     0x07
But9            Equ     0x08
ProgUp          Equ     d'16'
ProgDn          Equ     d'17'
VolUp           Equ     d'18'
VolDn           Equ     d'19'


        org     0x0000                  ;org sets the origin, 0x0000 for the 16F628,
        goto    Start                   ;this is where the program starts running

        org     0x005

Start           movlw   0x07
                movwf   CMCON                   ;turn comparators off (make it like a 16F84)

                clrf    IR_PORT                 ;make PortB outputs low


                bsf     STATUS,         RP0     ;select bank 1
                movlw   b'11111101'             ;set PortB all inputs, except RB1
                movwf   IR_TRIS
                movlw   0xff
                movwf   PORTA
                bcf     STATUS,         RP0     ;select bank 0

Read_Sw
                btfss   PORTA,  SW1
                call    Switch1
                btfss   PORTA,  SW2
                call    Switch2
                btfss   PORTA,  SW3
                call    Switch3
                btfss   PORTA,  SW4
                call    Switch4
                call    Delay27
                goto    Read_Sw

Switch1         movlw   ProgUp
```

```
                call    Xmit_RS232
                retlw   0x00

Switch2         movlw   ProgDn
                call    Xmit_RS232
                retlw   0x00

Switch3         movlw   VolUp
                call    Xmit_RS232
                retlw   0x00

Switch4         movlw   VolDn
                call    Xmit_RS232
                retlw   0x00

TX_Start        movlw   d'92'
                call    IR_pulse
                movlw   d'23'
                call    NO_pulse
                retlw   0x00

TX_One          movlw   d'46'
                call    IR_pulse
                movlw   d'23'
                call    NO_pulse
                retlw   0x00

TX_Zero         movlw   d'23'
                call    IR_pulse
                movlw   d'23'
                call    NO_pulse
                retlw   0x00

IR_pulse
                MOVWF   count           ;  Pulses the IR led at 38KHz
irloop          BSF     IR_PORT,        IR_Out
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                BCF     IR_PORT,        IR_Out
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP
                NOP
                NOP                     ;
                NOP                     ;
                DECFSZ  count,F
                GOTO    irloop
                RETLW   0

NO_pulse
                MOVWF   count           ;  Doesn't pulse the IR led
irloop2         BCF     IR_PORT,        IR_Out
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                BCF     IR_PORT,        IR_Out
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP
                NOP
                NOP                     ;
                NOP                     ;
                DECFSZ  count,F
                GOTO    irloop2
                RETLW   0

Xmit_RS232      MOVWF   Data_Byte               ;move W to Data_Byte
                MOVLW   0x07                    ;set 7 DATA bits out
                MOVWF   Bit_Cntr
                call    TX_Start                ;send start bit
Ser_Loop        RRF     Data_Byte , f           ;send one bit
                BTFSC   STATUS    , C
                call    TX_One
                BTFSS   STATUS    , C
                call    TX_Zero
                DECFSZ  Bit_Cntr  , f           ;test if all done
                GOTO    Ser_Loop
                                                ;now send device data
                movlw   D'1'
                movwf   Dev_Byte                ;set device to TV
                MOVLW   0x05                    ;set 5 device bits out
                MOVWF   Bit_Cntr
Ser_Loop2       RRF     Dev_Byte , f            ;send one bit
                BTFSC   STATUS    , C
                call    TX_One
                BTFSS   STATUS    , C
                call    TX_Zero
                DECFSZ  Bit_Cntr  , f           ;test if all done
                GOTO    Ser_Loop2
                retlw   0x00



;Delay routines

Delay255        movlw   0xff            ;delay 255 mS
                goto    d0
Delay100        movlw   d'100'          ;delay 100mS
                goto    d0
Delay50         movlw   d'50'           ;delay 50mS
                goto    d0
Delay27         movlw   d'27'           ;delay 27mS
                goto    d0
Delay20         movlw   d'20'           ;delay 20mS
                goto    d0
Delay5          movlw   0x05            ;delay 5.000 ms (4 MHz clock)
d0              movwf   count1
d1              movlw   0xC7
                movwf   counta
                movlw   0x01
                movwf   countb
Delay_0         decfsz  counta, f
                goto    $+2
                decfsz  countb, f
                goto    Delay_0

                decfsz  count1  ,f
                goto    d1
                retlw   0x00

;end of Delay routines

        end
```

Tutorial 5.3 - requires one Main Board, one IR Board and LED Board.

This program implements toggling the 8 LED's on the LED board with the buttons 1 to 8 on a Sony TV remote control, you can easily change the device ID and keys used for the LED's. I've also used a (so far unused) feature of the 16F628, the EEPROM data memory - by using this the program remembers the previous settings when unplugged - when you reconnect the power it restores the last settings by reading them from the internal non-volatile memory. The 16F628 provides 128 bytes of this memory, we only use one here (address 0x00, set in the EEPROM_Addr constant).

```
;Tutorial 5_3
;Read SIRC IR and toggle LED display, save settings in EEPROM data memory.
;Nigel Goodwin 2002

        LIST    p=16F628                ;tell assembler what chip we are using
        include "P16F628.inc"           ;include the defaults for the chip
        ERRORLEVEL      0,      -302    ;suppress bank selection messages
        __config 0x3D18                 ;sets the configuration settings (oscillator type etc.)




                cblock  0x20                    ;start of general purpose registers
                        count                   ;used in looping routines
                        count1                  ;used in delay routine
                        counta                  ;used in delay routine
                        countb                  ;used in delay routine
                        LoX
                        Bit_Cntr
                        Cmd_Byte
                        Dev_Byte
                        Flags
                        Flags2
                        tmp1                    ;temporary storage
                        tmp2
                        tmp3
                        lastdev
                        lastkey

                endc

LED_PORT        Equ     PORTB
LED_TRIS        Equ     TRISB

IR_PORT         Equ     PORTA
IR_TRIS         Equ     TRISA
IR_In           Equ     0x02                    ;input assignment for IR data

OUT_PORT        Equ     PORTB
LED0            Equ     0x00
LED1            Equ     0x01
LED2            Equ     0x02
LED3            Equ     0x03
LED4            Equ     0x04
LED5            Equ     0x05
LED6            Equ     0x06
LED7            Equ     0x07
```

```
EEPROM_Addr     Equ     0x00                    ;address of EEPROM byte used

ErrFlag         Equ     0x00
StartFlag       Equ     0x01                    ;flags used for received bit
One             Equ     0x02
Zero            Equ     0x03

New             Equ     0x07                    ;flag used to show key released

TV_ID           Equ     0x01                    ;TV device ID

But1            Equ     0x00                    ;numeric button ID's
But2            Equ     0x01
But3            Equ     0x02
But4            Equ     0x03
But5            Equ     0x04
But6            Equ     0x05
But7            Equ     0x06
But8            Equ     0x07
But9            Equ     0x08

                org     0x0000
                goto    Start

                org     0x0004
                retfie

Start           movlw   0x07
                movwf   CMCON                   ;turn comparators off (make it like a 16F84)

Initialise      clrf    count
                clrf    PORTA
                clrf    PORTB
                clrf    Flags
                clrf    Dev_Byte
                clrf    Cmd_Byte



SetPorts        bsf     STATUS,         RP0     ;select bank 1
                movlw   0x00                    ;make all LED pins outputs
                movwf   LED_TRIS
                movlw   b'11111111'             ;make all IR port pins inputs
                movwf   IR_TRIS
                bcf     STATUS,         RP0     ;select bank 0

                call    EE_Read                 ;restore previous settings

Main
                call    ReadIR                  ;read IR signal
                call    ProcKeys                ;do something with commands received

                goto    Main                    ;loop for ever

ProcKeys
                btfss   Flags2, New
                retlw   0x00                    ;return if not new keypress
                movlw   TV_ID                   ;check for TV ID code
                subwf   Dev_Byte,       w
                btfss   STATUS    , Z
                retlw   0x00                    ;return if not correct code

                movlw   But1                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key1                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED0            ;and test LED bit for toggling
                bsf     LED_PORT,       LED0    ;turn on LED
                btfsc   tmp3,   LED0
                bcf     LED_PORT,       LED0    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key1            movlw   But2                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key2                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED1            ;and test LED bit for toggling
                bsf     LED_PORT,       LED1    ;turn on LED
                btfsc   tmp3,   LED1
                bcf     LED_PORT,       LED1    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key2            movlw   But3                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key3                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED2            ;and test LED bit for toggling
                bsf     LED_PORT,       LED2    ;turn on LED
                btfsc   tmp3,   LED2
                bcf     LED_PORT,       LED2    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key3            movlw   But4                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key4                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED3            ;and test LED bit for toggling
                bsf     LED_PORT,       LED3    ;turn on LED
                btfsc   tmp3,   LED3
                bcf     LED_PORT,       LED3    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key4            movlw   But5                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key5                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED4            ;and test LED bit for toggling
                bsf     LED_PORT,       LED4    ;turn on LED
                btfsc   tmp3,   LED4
                bcf     LED_PORT,       LED4    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00
```

```
Key5            movlw   But6                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key6                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED5            ;and test LED bit for toggling
                bsf     LED_PORT,       LED5    ;turn on LED
                btfsc   tmp3,   LED5
                bcf     LED_PORT,       LED5    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key6            movlw   But7                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                goto    Key7                    ;try next key if not correct code

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED6            ;and test LED bit for toggling
                bsf     LED_PORT,       LED6    ;turn on LED
                btfsc   tmp3,   LED6
                bcf     LED_PORT,       LED6    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

Key7            movlw   But8                    ;test for button 1
                subwf   Cmd_Byte, w
                btfss   STATUS    , Z
                retlw   0X00

                movf    LED_PORT,       w       ;read PORTB (for LED status)
                movwf   tmp3                    ;and store in temp register
                btfss   tmp3,   LED7            ;and test LED bit for toggling
                bsf     LED_PORT,       LED7    ;turn on LED
                btfsc   tmp3,   LED7
                bcf     LED_PORT,       LED7    ;turn off LED
                bcf     Flags2, New             ;and cancel new flag
                call    EE_Write                ;save the settings
                retlw   0x00

EE_Read         bsf     STATUS, RP0             ; Bank 1
                movlw   EEPROM_Addr
                movwf   EEADR                   ; Address to read
                bsf     EECON1, RD              ; EE Read
                movf    EEDATA, W               ; W = EEDATA
                bcf     STATUS, RP0             ; Bank 0
                movwf   LED_PORT                ; restore previous value
                retlw   0x00

EE_Write        movf    LED_PORT,       w       ; read current value
                bsf     STATUS, RP0             ; Bank 1
                bsf     EECON1, WREN            ; Enable write
                movwf   EEDATA                  ; set EEPROM data
                movlw   EEPROM_Addr
                movwf   EEADR                   ; set EEPROM address
                movlw   0x55
                movwf   EECON2                  ; Write 55h
                movlw   0xAA
                movwf   EECON2                  ; Write AAh
                bsf     EECON1, WR              ; Set WR bit
                                                ; begin write
                bcf     STATUS, RP0             ; Bank 0

                btfss   PIR1,   EEIF            ; wait for write to complete.
                goto    $-1
                bcf     PIR1,   EEIF            ; and clear the 'write complete' flag
                bsf     STATUS, RP0             ; Bank 1
                bcf     EECON1, WREN            ; Disable write
                bcf     STATUS, RP0             ; Bank 0
                retlw   0x00



;IR routines

ReadIR          call    Read_Pulse
                btfss   Flags,  StartFlag
                goto    ReadIR                  ;wait for start pulse (2.4mS)

Get_Data        movlw   0x07                    ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Cmd_Byte
Next_RcvBit2    call    Read_Pulse
                btfsc   Flags,  StartFlag       ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,  ErrFlag         ;abort if error
                goto    ReadIR

                bcf     STATUS    , C
                btfss   Flags,  Zero
                bsf     STATUS    , C
                rrf     Cmd_Byte  , f
                decfsz  Bit_Cntr  , f
                goto    Next_RcvBit2

                rrf     Cmd_Byte  , f           ;correct bit alignment for 7 bits

Get_Cmd         movlw   0x05                    ;set up to read 5 bits
                movwf   Bit_Cntr
                clrf    Dev_Byte
Next_RcvBit     call    Read_Pulse
                btfsc   Flags,  StartFlag       ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,  ErrFlag         ;abort if error
                goto    ReadIR

                bcf     STATUS    , C
                btfss   Flags,  Zero
                bsf     STATUS    , C
                rrf     Dev_Byte  , f
                decfsz  Bit_Cntr  , f
                goto    Next_RcvBit

                rrf     Dev_Byte  , f           ;correct bit alignment for 5 bits
                rrf     Dev_Byte  , f
                rrf     Dev_Byte  , f

                retlw   0x00

;end of ReadIR


;read pulse width, return flag for StartFlag, One, Zero, or ErrFlag
;output from IR receiver is normally high, and goes low when signal received

Read_Pulse      clrf    LoX
                btfss   IR_PORT,        IR_In   ;wait until high
                goto    $-1
                clrf    tmp1
                movlw   0xC0                    ;delay to decide new keypress
                movwf   tmp2                    ;for keys that need to toggle

Still_High      btfss   IR_PORT,        IR_In   ;and wait until goes low
```

```
                goto    Next
                incfsz  tmp1,f
                goto    Still_High
                incfsz  tmp2,f
                goto    Still_High
                bsf     Flags2, New             ;set New flag if no button pressed
                goto    Still_High

Next            nop
                nop
                nop
                nop
                nop                             ;waste time to scale pulse
                nop                             ;width to 8 bits
                nop
                nop
                nop
                nop
                nop
                nop
                incf    LoX,    f
                btfss   IR_PORT,        IR_In
                goto    Next                    ;loop until input high again

; test if Zero, One, or Start (or error)

Chk_Pulse       clrf    Flags

TryError        movf    LoX,    w               ; check if pulse too small
                addlw   d'255' - d'20'          ; if LoX <= 20
                btfsc   STATUS    , C
                goto    TryZero
                bsf     Flags,  ErrFlag         ; Error found, set flag
                retlw   0x00

TryZero         movf    LoX,    w               ; check if zero
                addlw   d'255' - d'60'          ; if LoX <= 60
                btfsc   STATUS    , C
                goto    TryOne
                bsf     Flags,  Zero            ; Zero found, set flag
                retlw   0x00

TryOne          movf    LoX,    w               ; check if one
                addlw   d'255' - d'112'         ; if LoX <= 112
                btfsc   STATUS    , C
                goto    TryStart
                bsf     Flags,  One             ; One found, set flag
                retlw   0x00

TryStart        movf    LoX,    w               ; check if start
                addlw   d'255' - d'180'         ; if LoX <= 180
                btfsc   STATUS    , C
                goto    NoMatch
                bsf     Flags,  StartFlag       ; Start pulse found
                retlw   0x00
NoMatch                                         ; pulse too long
                bsf     Flags,  ErrFlag         ; Error found, set flag
                retlw   0x00

;end of pulse measuring routines



;Delay routines

Delay255        movlw   0xff            ;delay 255 mS
                goto    d0
Delay100        movlw   d'100'          ;delay 100mS
                goto    d0
```

```
Delay50         movlw   d'50'           ;delay 50mS
                goto    d0
Delay20         movlw   d'20'           ;delay 20mS
                goto    d0
Delay5          movlw   0x05            ;delay 5.000 ms (4 MHz clock)
d0              movwf   count1
d1              movlw   0xC7
                movwf   counta
                movlw   0x01
                movwf   countb
Delay_0         decfsz  counta, f
                goto    $+2
                decfsz  countb, f
                goto    Delay_0

                decfsz  count1  ,f
                goto    d1
                retlw   0x00

;end of Delay routines


                end
```

The EEPROM data is accessed by two new routines, EE_Read and EE_Write, the EE_Read routine is called as the program powers up, before we enter the main loop, and the EE_Write routine is called after every LED change. The EE_Read routine is very straightforward, we simply set the address we wish to read in the EEADR register, set the RD flag in the EECON1 register, and then read the data from the EEDATA register. Writing is somewhat more complicated, for a couple of reasons:

1. Microchip have taken great care to prevent accidental or spurious writes to the data EEPROM. In order to write to it we first have to set the 'Write Enable' bit in the EECON1 register, and then make two specific writes (0x55 and 0xAA) to the EECON2 register, only then can we set the WR bit in EECON1 and start the actual writing. One of the most common problems in domestic electronics today is data EEPROM corruption, hopefully the efforts of Microchip will prevent similar problems with the 16F628.
2. Writing to EEPROM takes time, so we have to wait until the 'Write Complete' flag is set, it doesn't really matter in this application as the time spent waiting for the next IR command gives more than enough time to write to the data EEPROM, but it's good practice to do it anyway.

The extra work involved makes the EE_Write routine a lot longer than the EE_Read routine, it also doesn't help that we need to access registers in different banks, so we do a fair bit of bank switching.