SIMPLE D.I.Y. DMX-512 CONTROLLER

INTRODUCTION

This document describes the design and construction of a very simple DMX controller.

DMX is the technology used in theatres to control the lighting. The theory of DMX is not explained here, as that information is freely available on the internet.

This design is offered for anyone to use, entirely at their own risk.

SPECIFICATION

The device controls one or more LED lanterns. It can be switched to control lanterns set to either 4-channel mode, or 7-channel mode. The software included in this document allows the device to control up to 12 lanterns (or groups of lanterns); each having a unique starting address.

The design could easily be modified to control a different number of groups of lanterns by a simple modification to the program – see appendix 3.

The addresses used by each lantern would be as follows:

Lantern	4-Channel Mode	7-Channel Mode
1	1 – 4	1 – 7
2	5 – 8	8 – 14
3	9 – 12	15 – 21
:	:	:
n	(4n-3) to (4n).	(7n-6) to (7n)

The cost of the finished device is around £50 (2018 prices).

Illustration 1: View of the top of the enclosure.

[Note: This is an earlier version that does not have the 4/7 switch]

CONTROL INTERFACE

The device uses 4 dimmers for the colours and a master dimmer to facilitate fades and blackouts. It does not control the strobe or special effect channels, even for lanterns set to 7 channel mode.

A switch selects 4-channel or 7-channel mode.

CIRCUIT DESIGN

The circuit is as shown:

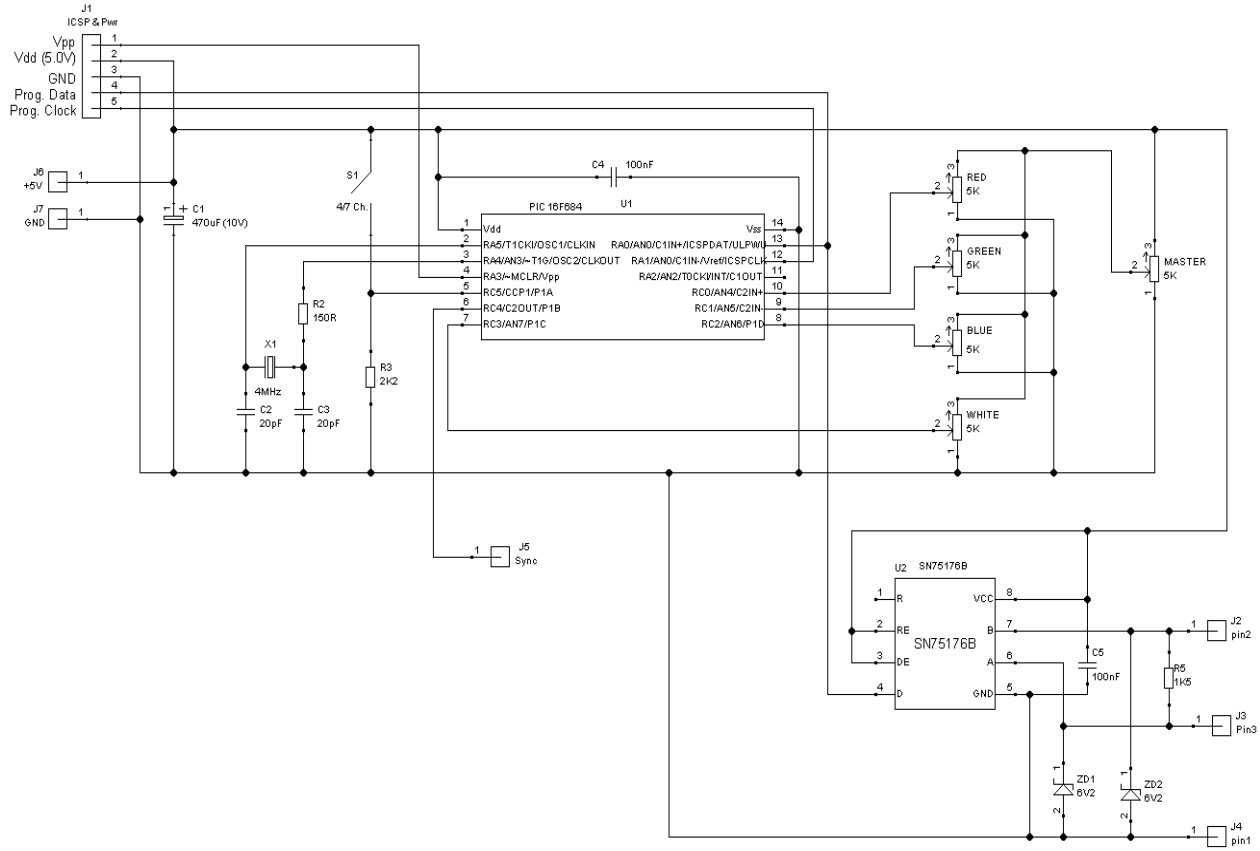


Illustration 2: Schematic Circuit Diagram

The main parts to the circuit are:

1) Microprocessor.
2) Output driver.
3) Faders (slider potentiometers).
4) 4/7 Channel Switch
5) In-Circuit Serial Programming Interface.
6) DMX output socket.
7) Sync Output
8) Power Supply

MICROCONTROLLER

The heart of the device is a PIC Microprocessor. This is a complete 'computer on a chip'. The design uses a PIC16F684 chip, but any of the PIC family of processors with 4 ADCs will do. If a different chip is used, it is essential to make the necessary changes to the program code, especially the configuration bits.

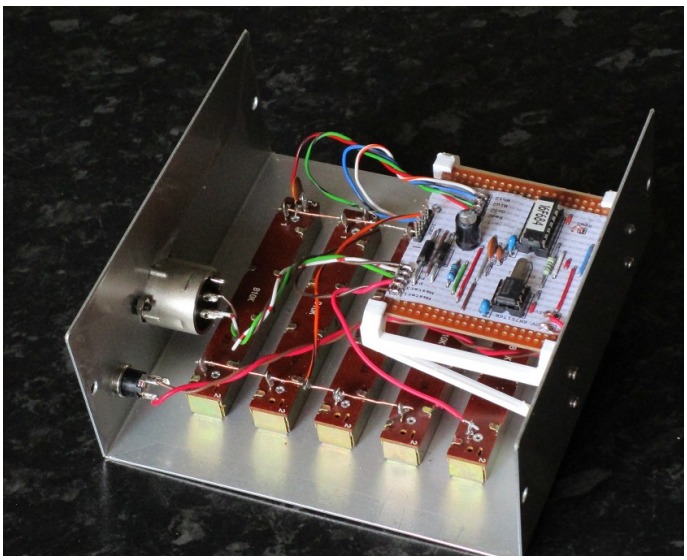


Illustration 3: Layout of components inside the enclosure

[Note: This is an earlier version that does not have the 4/7 switch]

The timing requirements for DMX are fairy strict, so a crystal controlled oscillator is used to keep well within the spec.

OUTPUT DRIVER

The SN761768 is a chip very commonly used in the input or output stage of network devices. It turns the digital signal from the microprocessor into a balanced-line signal, with sufficient power to drive up to 30 similar devices. Applying a voltage to the RE or DE pins determines whether it works as an input or output device. In this circuit it is hard-wired for output only.

FADERS

There are 5 slider potentiometers (faders). The master fader is wired across the supply rails, so that the output is a voltage between 0V and 5V proportional to the position of the slider. This voltage then supplies the colour faders so that each colour is a proportion of the master. The colour sliders are connected to the Analogue-to-Digital inputs of the microprocessor.

There is a small problem with the direct connection of the master fader to the colour faders; it introduces a slight non-linearity to the master voltage due to the load imposed by the colour faders. In practice this proved acceptable for most uses, however, if strict linearity is required this can be obtained by either an analogue or digital fix. The analogue fix is to buffered the master output, using, for example, an op-amp. The digital solution is to connect the master independently to an input of the processor, and do the scaling using arithmetic in the program.

4/7 CHANNEL SWITCH

A switch and resistor connected to pin 5 of the microprocessor determines whether the output is configured for 4-channel mode or 7-channel mode. The pin is polled at the start of every DMX frame, so can be changed 'in flight'.

IN-CIRCUIT SERIAL PROGRAMMING INTERFACE

When working with microprocessors it is always useful to be able to re-program them without physically unplugging them from the circuit. The PIC chips can be re-programmed in-circuit by providing a suitable connection, know as the in-circuit serial programming (ICSP) interface. Note that during programming the 5V power is provided by the programming device, so the normal power supply should not be connected. The fact that pin 13 is used for both the ICSP data and the DMX out signal does not matter. Programming is not affected by the ICSP signal also going to the

input of the line driver, but it is a wise precaution to unplug anything connected to the DMX output during programming.

DMX SOCKET

The DMX output interface is a female XLR socket, either 3-pin or 5-pin, whichever is more compatible with your other equipment (a 3-pin socket was used in this case). In either case, the pin assignments are the same: Pin 1 is the common 0V line (usually the screen of the cable), pin 2 carries the Data- signal (usually the white or blue conductor) and pin 3 carries the Data+ signal (the red conductor). Note that pin 1 should not be connected to the chassis or earth, but the body of the socket can be earthed.

The 1.5K resistor and the two Zener diodes are to protect the circuit from accidental over-voltages on the DMX cable.

Illustration 4: Side view showing the power and DMX connections

[Note: This is an earlier version that does not have the 4/7 switch]

SYNC OUTPUT

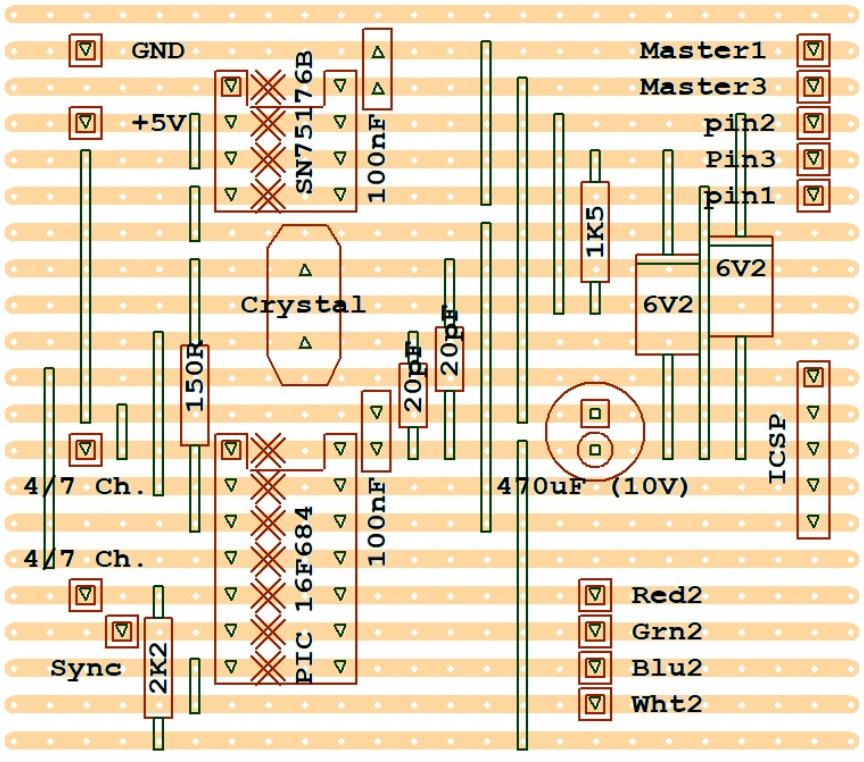
It is useful to be able to view the DMX output on an oscilloscope or datascope. A trigger pulse is produced at the start of each DMX Frame to initiate the 'scope sweep. Pin 6 is used as the output, programmed to produce a positive-going pulse during the MAB period.

POWER SUPPLY

Any regulated 5V supply with adequate output will be suitable. The big electrolytic across the supply on the circuit board is necessary to ensure reliable operation. The power connector is a socket that matches the power adapter. Note that while programming the device, the power adapter should be unplugged.

CIRCUIT BOARD

The circuit is build on a piece of stripboard. laid out as shown below.



It is supported inside the box by 3D-printed brackets.

MICROPROCESSOR PROGRAM

The PIC chip was programmed using the MPLAB IDE package.

The program design starts with the structure of the basic DMX protocol. See Appendix 1.

The program design needs to accommodate the repetition of the four colour levels for 12 lanterns, so the basic structure is elaborated as shown in Appendix 2. The channel usage for the two different configurations are:

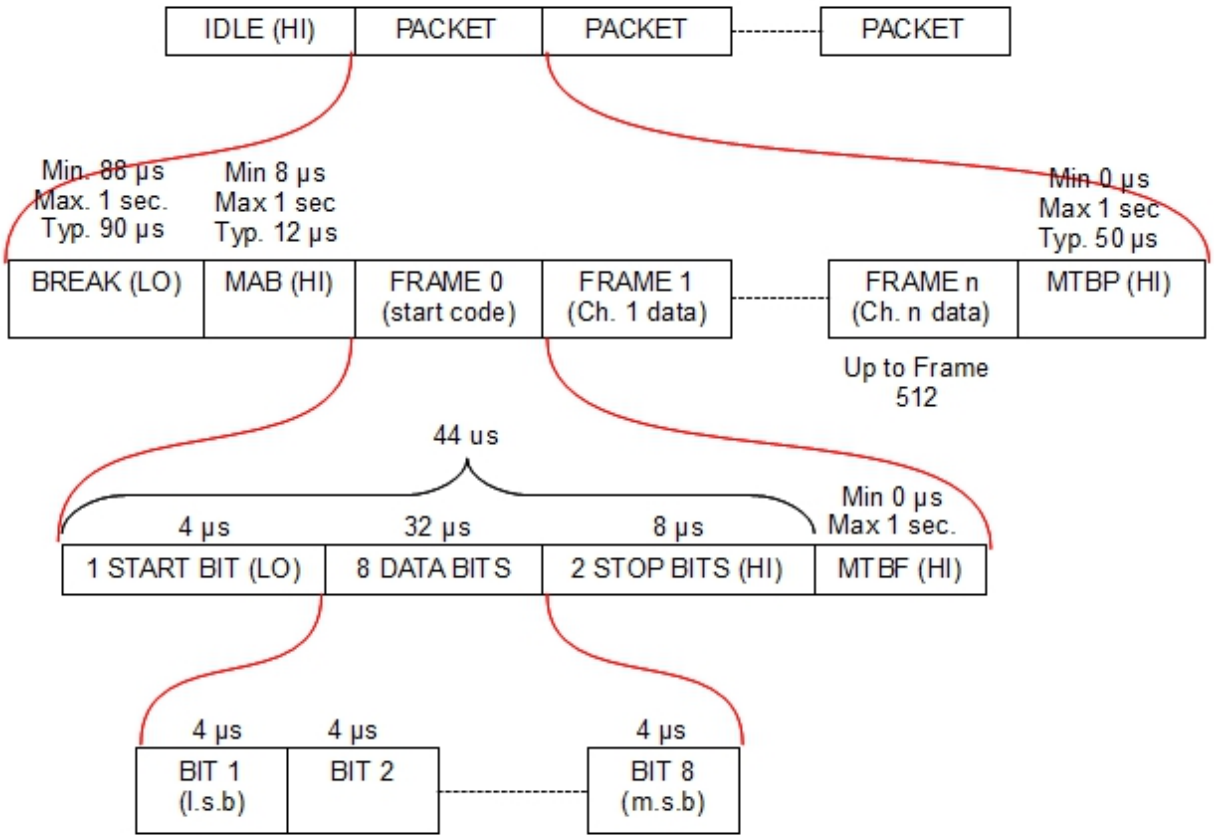
Channel	4-channels	7-channels
1	Red	Master dimmer. (Always 255)
2	Green	Strobe. (Always 0)
3	Blue	Red
4	White	Green
5		Blue
6		White
7		Special functions. (Always 0)

The assembler code is shown in Appendices 5 and 6. The code is heavily commented so a detailed explanation is not given here, but one subtle point to keep in mind is that when the processor executes an instruction that alters the state of an output pin, the change does not occur until the end of the instruction. This is important when timing is critical.

APPENDIX 1 - Standard DMX Protocol and Timing

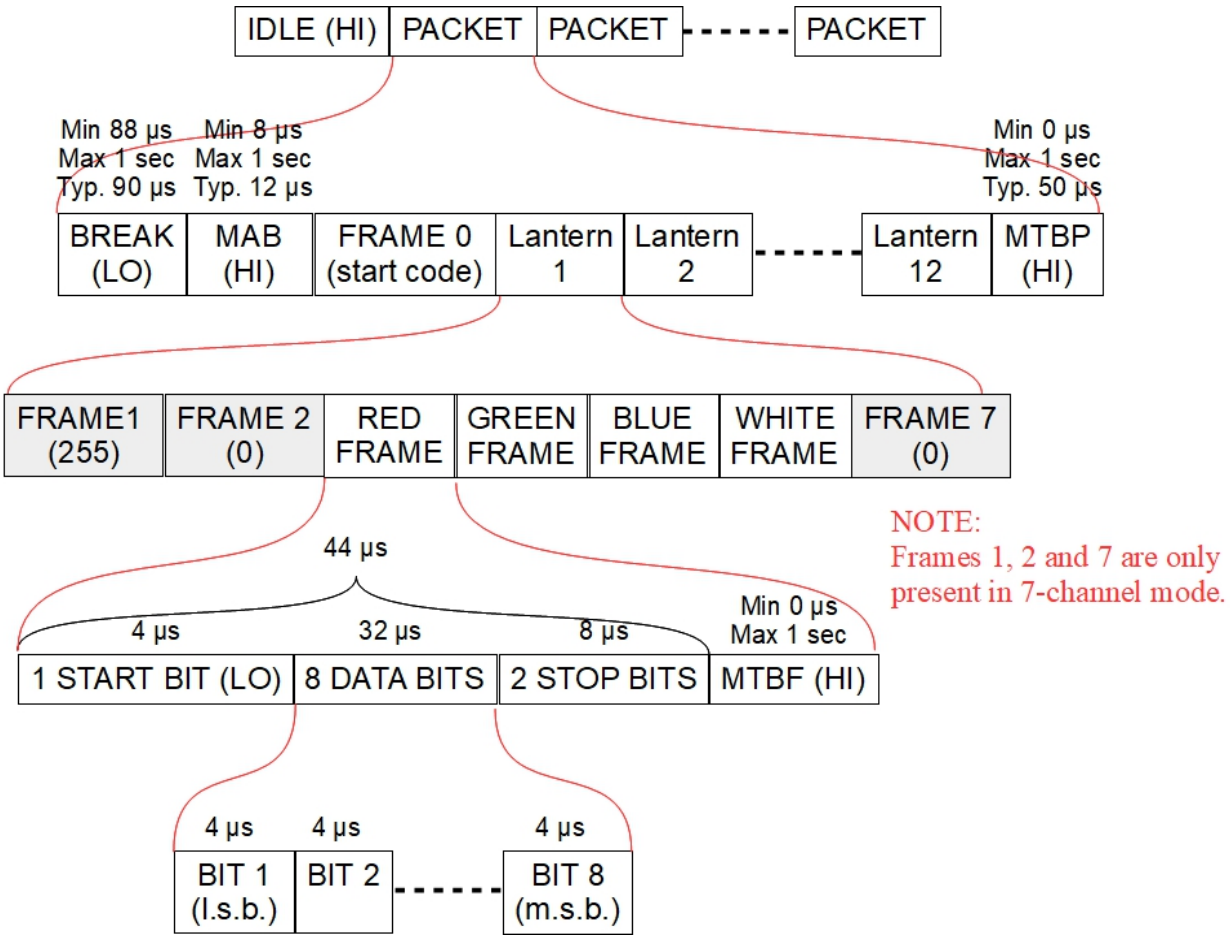
Data rate is 250,000 bits per second.
1 DMX Bit occupies 4 micro-second. (µs)
With a PIC Microprocessor, 1 instruction takes 4 clock cycles
With a 4MHz processor 1 Instruction takes 1 µs
Therefore 1 DMX Bit = 4 instructions

Timings below are in Microseconds (µs) or instructions



Minimum BREAK to BREAK time is 1204uS (i.e. At least 24 frames at minimum inter frame timings)

APPENDIX 2 - Elaborated DMX Structure



APPENDIX 4 - Program Code

```
;   Prog for controlling DMX lights using PIC16F684                  *
;                                                                    *
;   Reads voltages on 4 potentiometers and outputs same 4 DMX        *
;   Channels to 12 lanterns.                                         *
;   Switchable between lanterns in 4-Channel or 7-Channel mode.      *
;   Channels are:                                                    *
;   Ch. No.  4-Ch Mode   7-Ch Mode                                   *
;      1     Red         Master (always 255)                         *
;      2     Green       Strobe (always 0)                           *
;      3     Blue        Red                                         *
;      4     White       Green                                       *
;      5     -           Blue                                        *
;      6     -           White                                       *
;      7     -           Special Effects (always 0)                  *
;                                                                    *
;   A sync pulse, suitable for triggering an oscilloscope at the     *
;   start of each Packet is provided                                 *
;                                                                    *
;   Static (not re-locatable) program.                               *
;                                                                    *
;    Filename:      SimpleDMX                                        *
;    Author:        Terry Powderhill                                 *
;    Company:       Home                                             *
;    Copyright:     None.  Free for anyone to use at their own risk  *
;                                                                    *
;    Files required: 16f684.lkr, P16F684.INC                         *
;                                                                    *
;    Pin Assignment:                                                 *
;                          Vdd = 1   14 = Gnd                        *
;                     RA5/OSC1 = 2   13 = RA0 DMX Out/ICSP Data      *
;                     RA4/OSC2 = 3   12 = RA1/ICSP Clock             *
;                      RA3/Vpp = 4   11 = RA2/AN2 (not used)         *
;       RC5 4/7 Channel switch = 5   10 = RC0/AN4 Red in             *
;    RC4 'Scope sync pulse out = 6    9 = RC1/AN5 Green in           *
;              RC3/AN7 White in = 7    8 = RC2/AN6 Blue in           *
;   To simplify programming, the only I/O on PortA is the DMX output *
;      All other I/O is on PortC                                     *
;   Interrupts are not used (disabled)                               *
;   External 4MHz clock used to ensure accurate timing               *
;*********************************************************************

       list      p=16F684          ; list directive to define processor
       #include <P16f684.INC>      ; processor specific variable definitions

       errorlevel  -302            ; suppress message 302 from list file

;   '__CONFIG' directive is used to embed configuration word within .asm file.
;   The labels are defined in the .inc file.  Must be listed in ascending order
;   Config switches:
;       _BOD_OFF             - H'3CFF' Brown-out Detection off
;       _MCLRE_OFF           - H'3FDF' GP3 pin not used as MCLR
;       _PWRTE_ON            - H'3FEF' Power-up Timer on
;       _WDT_OFF             - H'3FF7' Watchdog Timer off
;       _HS_OSC              - H'3FF9' Crystal oscillator at <3.5 MHz
;       _EXTRC_OSC_NOCLKOUT - H'3FFE' External Oscillator, no clock out
;       _CP_OFF              - H'3FFF' Code Protection off
;       _CPD_OFF             - H'3FFF' Data Code Protection off

       __CONFIG  _BOD_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC & _EXTRC_OSC_NOCLKOUT &
_CP_OFF & _CPD_OFF

;*****************************************************************************
;Defines                                                                    *
;*****************************************************************************
;Ports bit assignment
#define Snc_Out PORTC,0x04     ;RC<4> Oscilloscope sync output pin
#define Dmx_Out  PORTA,0x00    ;RA<0> DMX output pin
#define Mode_in PORTC,0X05  ;RC<5> HI = 4-ch Mode, LO = 7-ch mode

;*****************************************************************************
;General Purpose Registers (GPR's), starting at address 20h                 *
;*****************************************************************************
    cblock  0x20
    RED_VAL       ; Red DMX value
    GRN_VAL       ; Green DMX value
    BLU_VAL       ; Blue DMX value
```

```
    WHT_VAL       ; White DMX value
    TEMP          ; Used for rotating output values
    LC            ; General Loop counter
    LEDLP         ; Loop counter for Lanterns
    endc

;*****************************************************************************
;Reset Vector.   Jumps to here on power up or after reset                    *
;*****************************************************************************
        ORG   0x000    ; processor reset vector
        nop            ; required by in circuit debugger
        goto  MAIN     ; go to beginning of program

;*****************************************************************************
;Interrupt Vector.  Jumps to here after an interrupt                         *
; Interrupts are disabled, but include this in case we want                  *
; to use interrupts in the future                                            *
;*****************************************************************************
        ORG    0x004
        return         ; interrupt trap - returns without re-enabling

;*****************************************************************************
;Initialization                                                              *
; Start main program at address h10                                          *
;*****************************************************************************
MAIN    org   0x0010

;*****************************************************************************
;Settings for special purpose registers                                      *
; No settings required for:                                                  *
;  PIR1 (Peripheral Interrupt Register 1)                                    *
;  OPTION_REG (Options)                                                      *
;  PIE1 (Peripheral Interrupt Enable 1)                                      *
;  OSCAL (Oscillator Calibration, because using external osc.)               *
;*****************************************************************************
;Registers in Bank 0 (or both)..
        BANKSEL PORTA    ;select bank 0

;*****************************************************************************
;INTCON (Interrupt Control)                                                  *
;*****************************************************************************
        clrf   INTCON    ;Disable all interrupts by clearing all bits

;*****************************************************************************
;CMCON (Comparator Control)                                                  *
;n/a COUT n/a CINV CIS CM2 CM1 CM0                                          *
; 0   0    0   0    0   1   1   1                                           *
;set CM<2:0> = 111 to disconnect comparator from I/O ports                   *
;*****************************************************************************
        movlw   b'00000111'
        movwf   CMCON0

;*****************************************************************************
;ADCON0 (A/D Control)                                                        *
; ADFM VCFG n/a CHS2 CHS1 CHS0 GO ADON                                       *
; left Vdd   0   ?    ?    ?   0   1                                        *
; Output is 10 bits, placed in ADRESH and ADRESL, either left or right       *
; shifted, controlled by ADFM (ADCON<7>)                                     *
; set ADFM = 1 for left shift (i.e. Most significant 8 bits in ADRESH).      *
; The 2 LSB in ADRESL will be ignored for our purposes.                      *
; There are 2 options for voltage reference, controlled by VCFG (ADCON<6>).  *
; 0=Vdd, 1=Vref.                                                             *
; There are 8 A/D Channels: AN0 to AN7.  CHS2, CHS1 and CHS0 bits            *
; (ADCON<4:2>) select which channel is connected to the sample and hold      *
; buffer.  Conversion is started by setting GO/DONE (ADCON<1>) to 1. (gets   *
; set to 0 when conversion is finished).                                     *
; ADON Turns on A/D conversion capability.  We'll select the ports later     *
;*****************************************************************************

;Registers in Bank 1
        BANKSEL   TRISA    ;select bank 1

;*****************************************************************************
;ADCON1 (A/D Control)                                                        *
;n/a ADCS2 ADCS1 ADCS0 n/a n/a n/a n/a                                       *
; 0    0     0     1    0   0   0   0                                       *
; The Clock Source for the AD Converter is derived from the device frequency, *
; controlled by ADCS2:ADCS0. For 4MHz processor the values should be 001, and *
; conversion is complete in 2uS.                                             *
;*****************************************************************************
```

```
        movlw   b'00010000'
        movwf   ADCON1

;*********************************************************************************
;TRISA (PORTA Tri-State)                                                         *
;n/a n/a RA5   RA4      RA3      RA2     RA1        RA0                          *
;        OCS1  OSC2   NotUsed  NotUsed ICSPCLK DMXOut/ICSPD                     *
;0   0  input input    input    input   input      output                       *
;*********************************************************************************
        movlw   b'00111110'
        movwf   TRISA

;*********************************************************************************
;TRISC (PORTC Tri-State, same bank as TRISA)                                     *
;n/a n/a   RC5      RC4     RC3     RC2     RC1     RC0                         *
;        4/7-Mode SyncOut AN7-Wht AN6-Blu AN5-Grn AN4-Red                       *
;0   0    input    output  Input   Input   Input   Input                        *
;*********************************************************************************
        movlw   b'00101111'
        movwf   TRISC

;*********************************************************************************
;ANSEL (Analog Select) - Digital(0) or Analog(1) input?                          *
;ANS7 ANS6 ANS5 ANS4 ANS3 ANS2 ANS1 ANS0                                         *
; 1    1    1    1    0    0    0    0                                         *
;*********************************************************************************
        movlw   b'11110000'
        movwf   ANSEL

;*********************************************************************************
;Main program                                                                    *
; For timing; with 4MHz clock, a 1-instruction-cycle operation takes 1 micro    *
; second (uS).  Timing for Ops taking >1 cycle is shown thus: [2]               *
; 1 DMX bit is 4 micro second (4 single-cycle ops)                              *
; Note that output is not changed until the END of a set/clear instruction,     *
; so actual DMX output lags by 1 cycle.  Actual DMX output is shown in {}       *
;*********************************************************************************

        BANKSEL PORTA     ;select bank 0
;Start with Sync output LO and DMX HI (MTBF)
        bcf   Snc_Out

;We're allowed an idle period of HI output, but we won't bother.
;Process Packets forever
PACKET  bcf   Dmx_Out        ;Start of BREAK. LO for at least 88 uS. {end of MTBP}
;We'll use the time to do the A/D conversion.  min 34 cycles per colour
;Red value
        movlw   b'00010001'    ;select AN4 {LO = BREAK continues}
        call    ADC            ;[2+30]
        movwf   RED_VAL        ;Save Red value
;Green value
        movlw   b'00010101'    ;select AN5 {LO = BREAK continues}
        call    ADC
        movwf   GRN_VAL        ;Save Green value
;Blue value
        movlw   b'00011001'    ;select AN6 {LO = BREAK continues}
        call    ADC
        movwf   BLU_VAL        ;Save the Blue value
;White value
        movlw   b'00011101'    ;select AN7 {LO = BREAK continues}
        call    ADC
        movwf   WHT_VAL        ;Save the White value
;Have done at least 4 x 34 = 136 cycles, which exceeds required 88 so no more delay required.

;MARK AFTER BREAK.  min 8uS.  We'll do 11.  Also start the Sync pulse
        bsf     Snc_Out
        bsf     Dmx_Out        ;Output HI {LO = end of BREAK}
        movlw   d'2'           ;2 loops {HI = MAB}
        call    DELAY          ;[2+9] {HI = MAB}

;End the Sync pulse
        bcf     Snc_Out        ;{HI = MAB}

;START CODE - a 'dummy' Frame with all LO data bits
        clrw
        call    FRAME
;Process 12 lanterns of 4 or 7 Frames each. {HI = MTBF}
        movlw   d'12'          ;lantern loop
        movwf   LEDLP          ;keep count of Lanterns
;Output the 4 colours for the Lantern
```

```
LED     BTFSS   Mode_in         ;test 4/7 switch (HI = 4-ch mode)
        goto    RED             ;Skip MASTER channel in 4-ch mode
        movlw   d'255'          ;MASTER Channel = 255
        call    FRAME           ;[2]
        clrw                    ;Strobe channel = 0
        call    FRAME           ;[2]
RED     movf    RED_VAL,0       ;Move Red value to W.
        call    FRAME           ;[2]
        movf    GRN_VAL,0
        call    FRAME
        movf    BLU_VAL,0
        call    FRAME
        movf    WHT_VAL,0
        call    FRAME
        BTFSS   Mode_in         ;test 4/7 switch (HI = 4-ch mode)
        goto    NOEFF           ;Skip SPECIAL EFFECT channels in 4-ch mode
        clrw
        call    FRAME           ;[2] Special effect Channel = 0
NOEFF   bcf     STATUS,Z        ;clear the zero flag
        decfsz  LEDLP,1         ;Done all Lanterns?
        Goto    LED             ;[2] No: Do another Lantern
        movlw   d'15'           ;Yes: finish this packet
        call    DELAY           ;about 50 cycle Mark Time Between Packets
        goto    PACKET          ;[2] next Packet

;*****************************************************************************
;Routine to output one byte of DMX data previously loaded into W.             *
; We haven't got time to test and set individual bits in the 4 ops allowed    *
; for each bit, so output the whole byte to PORTA, relying on the fact that   *
; only RA0 is set to output.                                                  *
;*****************************************************************************
FRAME   movwf   TEMP            ;ready to read its bits
        bcf     Dmx_Out         ;1 start-bit LO {HI = end of MTBF}
        nop                     ;{LO = start of start-bit
        nop
        nop
        movwf   PORTA           ;Output bit0 {LO = end of start-bit}
        rrf     TEMP,1          ;Rotate right {start of bit0}
        movf    TEMP,0          ;move the rotated value to W {Continue bit0}
        nop                     ;{continue bit 0}
        movwf   PORTA           ;Output bit1 {end of bit 0}
        rrf     TEMP,1          ;{start of bit 1}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit2 {end of Bit 1}
        rrf     TEMP,1          ;{start of bit 2}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit3 {end of Bit 2}
        rrf     TEMP,1          ;{Start of bit 3}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit4 {end of Bit 3}
        rrf     TEMP,1          ;{start of bit4}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit5 {end of Bit 4}
        rrf     TEMP,1          ;{start of bit5}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit6 {end of bit 5}
        rrf     TEMP,1          ;{start of bit6}
        movf    TEMP,0
        nop
        movwf   PORTA           ;Output bit7 {end of Bit 6}
        rrf     TEMP,1          ;{start of bit 7}
        movf    TEMP,0
        nop
        bsf     Dmx_Out         ;[1] Output HI {end of bit 7}
        nop                     ;{HI = start of stop-bit 1}
        nop
        nop
        nop                     ;{HI = end of stop-bit 1}
        nop                     ;{HI = start of stop-bit 2}
        nop
        nop
        nop                     ;{HI = end of stop-bit 2
        return                  ;[2] {HI = Start of MTBF}
;Any further delay before the next frame will count as Mark Time Between Frames (MTBF)
```

```
;*******************************************************************************
;Routine to introduce a delay.  Load no of cycles into W before calling       *
;Delay is 3 x (W+1)                                                           *
;*******************************************************************************
DELAY  bcf     STATUS,Z       ;clear the zero flag
       movwf   LC             ;Loop Counter
       decfsz  LC,1           ;[2 if skips] Decrement LC, skip over next op if zero
       goto    $-1            ;Loop back [2]
       return                 ;[2]

;*******************************************************************************
;Do AD conversion. Load ADCON0 value for Port selection into W before calling *
;Takes about 30 cycles                                                        *
;*******************************************************************************
ADC    movwf   ADCON0         ;Select the A/D port
       movlw   d'5'           ;Allow 20uS for sample & hold acquisition
       call    DELAY          ;
       bsf     ADCON0,GO_DONE ;Start AD Conversion
       btfsc   ADCON0,GO_DONE ;Finished?
       goto    $-1            ;No: Check again
       movf    ADRESH,0       ;Yes: Result is returned in W
       return                 ;[2]

       END                    ; directive 'end of program'
```