AtomSoftTech – C18 Tips & Tricks

Basic Info:

Hello all, my name is Jason Lopez. I am the creator of this document and am also known as AtomSoft at various locations. I have created this document to remind myself of certain things in the C18 programming language. I have a bad memory and seem to forget a lot so I was thinking why not make a cheat sheet that I can print and use when i program. Almost all the code in this booklet was compiled on a PIC18F2525 or PIC18F4525.

Data Types and Limits

Type	Size	Minimum	Maximum
Char (1,2)	8 bits	-128	127
signed char	8 bits	-128	127
unsigned char	8 bits	0	255
int	16 bits	-32,768	32,767
unsigned int	16 bits	0	65,535
short	16 bits	-32,768	32,767
unsigned short	16 bits	0	65,535
short long	24 bits	-8,388,608	8,388,607
unsigned short long	24 bits	0	16,777,215
long	32 bits	-2,147,483,648	2,147,483,647
unsigned long	32 bits	0	4,294,967,295

Bitwise Operators

AND	OR	XOR	NOT	RIGHT SHIFT	LEFT SHIFT
&	|	^	~	>>	<<

Rational and Logical Operators

Equal To	Not Equal To	Greater Than	Greater Than or Equal To	Less Than	Less Than or Equal To	And	Or	Not
==	!=	>	>=	<	<=	&&	||	!

Interrupt (ISR)

High ISR

```
#pragma code high_vector = 0x08
void high_interrupt (void){
_asm goto high_ISR _endasm
}

#pragma code
#pragma interrupt high_ISR
void high_ISR (void) {
        // High ISR Code Here
}
```

Low ISR

```
#pragma code low_vector = 0x18
void low_interrupt (void){
_asm goto low_ISR _endasm
}

#pragma code
#pragma interrupt low_ISR
void low_ISR (void) {
        // Low ISR Code Here
}
```

AtomSoftTech – C18 Tips & Tricks

Program Flow

If	*If-else*	*Else-if*	*For*

`if(expression)`
`  statement;`
	`if(expression)`
	`  statement1;`
	`else`
	`  satement2;`
		`if(expression1)`
		`  statement1;`
		`else if (expression2)`
		`  satement2;`
		`else`
		`  statement3;`
			`for (expr1;expr2;expr3)`
			`  statement;`

While	*Switch*	*Do-While*

`while (expression)`
`  statement;`
	`switch (expression){`
	`  case const_expr1:`
	`  statement1;`
	`  break;`
	`  case const_expr2:`
	`  statement2;`
	`  break;`
	`  …`
	`  default:`
	`  statementn;`
	`}`
		`do`
		`  statement;`
		`while (expression);`

Main Useful

Header for Auto PIC	Define	Pragma's	Inline ASM
`#include <p18cxxx.h>`	`#define label constant`	`#pragma code [overlay]` `[section-name] [location]`	`_asm … _endasm`

Bit Set/Clear

Code	Example	Outcome
`#define bitset(var,bitno) (var|=1<<bitno)` **`#define bitclr(var,bitno) (var&=~(1<<bitno))`**	*`#define SpecialBit 2`* *`unsigned char MyVar;`* *`MyVar = 0;`* *`bitset(MyVar,SpecialBit);`*	*`MyVar would be 0 to start then after the bitset command, MyVar equals 00000100.`*

Delays – <delays.h>

Delay 1 Cycle aka NOP	Delay 10 Cycles	Delay 100 Cycles	Delay 1,000 Cycles	Delay 10,000 Cycles
`Delay1TCY();`	`Delay10TCYx(n);`	`Delay100TCYx(n);`	`Delay1KTCYx(n);`	`Delay10KTCYx(n);`

**NOTE: (n) = char only! (0-255)*

PORTB - <portb.h>

ClosePORTB	CloseRBxINT	OpenPORTB	OpenRBxINT
Disable the interrupts and internal pull-up resistors for PORTB.	Disable the interrupts for the specified INTx pin.	Configure the interrupts and internal pull-up resistors on PORTB.	Enable interrupts for the specified INTx pin.

`void ClosePORTB( void );`	`void CloseRB0INT( void );`
	`void CloseRB1INT( void );`
	`void CloseRB2INT( void );`
		`void OpenPORTB( unsigned char config);`	`void OpenRB0INT( unsigned char config );`
			`void OpenRB1INT( unsigned char config );`
			`void OpenRB2INT( unsigned char config );`

**NOTE: More information here "C:/MCC18/doc/periph-lib/IO ports.htm"*

AtomSoftTech – C18 Tips & Tricks

Using Pointers

Using a Pointer: Get a String of characters

```
void main(void)
{
    GetString((unsigned rom char*)"  AtomSoftTech");    //Send the string AtomSoftTech as a Rom Char to the function
}

void GetString(unsigned rom char *datapointer)          //The data is pointed to, using *datapointer rom char.
{
        char CurrentCharacter;                      //Setup a 1 char buffer

        while(*datapointer != 0){                   //While the data pointed to isnt a 0x00 do below
            CurrentCharacter = *datapointer++;      //put the data pointed to into CurrentCharacter buffer
        }
}
```

Using a Pointer: Get 8 bits and shift them into a variable

```
unsigned char MyString;
void main(void)
{
    SetBits(MyString);                  //Call the SetBits(*pointer) function with reference to MyString
}

void SetBits(unsigned char *datapointer)
{
    char x;                             //Loop Counter
    char tempData;                      //Temp Data

    for(x=0;x<8;x++){                   //Loop 8 Times
        tempData >>= 1;                 //Shift over bits to the right by 1

        if(MyPin);                      //If MyPin is high
            tempData |= 0x80;           //then OR in a 1 if not then leave it a zero

        Delay10KTCYx(100);              //Delay a short time
    }
    *datapointer = tempData;            //Loop is done so send out tempData to the pointer replacing it
}
```

Using EEPROM - <eep.h>

Storing and Reading Data from Internal EEPROM

```
#include <p18Cxxx.h>
#include <eep.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

// Just some variables to hold our data
unsigned char temp;
unsigned char address;
unsigned char data;

void main (void)
{
    OSCCON = 0x72;                          //8MHz clock
    while(!OSCCONbits.IOFS);                //Wait for OSC to become stable

    address = 0x00;                         //Set Address to 0
    data = 0x31;                            //Set Data to 0x31
    Write_b_eep (address, data);            //Write single byte to Internal EEP
    Busy_eep ();                            //Checks & waits the status of ER bit in EECON1 register
    temp = Read_b_eep (address);            //Read single byte from Internal EEP and store in temp
    while(1);                               //Loop Forever
}
```

Using a 74(LS/HC)164 - 8-Bit Serial In/Parallel Out Shift Register

Shifting data out MSB/LSB First into a 74xx164 - Also Shifting a Single Bit

```
#include <p18Cxxx.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

void ShiftByte(unsigned char data, char MLSBF);
void ShiftBit(char myBit);

#define SCL LATBbits.LATB1          //Clock Latch
#define SDO LATBbits.LATB0          //Data Latch

#define SCLT TRISBbits.TRISB1       //Clock TRIS
#define SDOT TRISBbits.TRISB0       //Data TRIS

void main (void)
{
    char x,tmp;                 //Some Variables
    OSCCON = 0x72;              //8MHz clock
    while(!OSCCONbits.IOFS);    //Wait for OSC to become stable

    SCLT = 0;                   //Set Clock as output
    SDOT = 0;                   //Set Data as output

    ShiftByte(0x00,0);          //Send all zeros to 164 (essentially clearing it)
    while(1){
        ShiftBit(1);            //Shift in a One

        for(x=0;x<7;x++){       //Loop 7 times
            ShiftBit(0);        //Shift in a Zero
        }

    }
}
void ShiftBit(char myBit){
            SDO = myBit;        //Set the pin high/low depending on user
            SCL = 1;            //Set the clock high
            Delay10TCYx(10);    //small delay
            SCL = 0;            //set the clock low to trigger complete bit
}
void ShiftByte(unsigned char data, char MLSBF)
{
    char tmp,x;
        for(x=0;x<8;x++){
            SDO = 0;                            //Prepare the port with a 0
            if(MLSBF == 1) tmp = data & 0x80;   //if we are using MSB First then AND with a 0x80
            if(MLSBF == 0) tmp = data & 0x01;   //if we are using LSB First then AND with a 0x01

            if(tmp != 0)            //If it is not a 0 then its a 1
                SDO = 1;            //if its a 1 set the data pin to a 1 if not it will stay a 0

            SCL = 1;                //set the clock high
            Delay10TCYx(10);        //small delay
            SCL = 0;                //set the clock low to trigger complete bit

             if(MLSBF == 1) data <<= 1;         //if using MSB First then SHIFT LEFT 1
             if(MLSBF == 0) data >>= 1;         //if using LSB First then SHIFT RIGHT 1
        }
}
```

Using ADC - <adc.h>

Setting up the ADC and Collecting data from AN0 (Using ADC Header file) A bit more confusing... Part 1 of 2

```
#include <p18cxxx.h>
#include <adc.h>
#include <delays.h>
 #pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

unsigned int result;
 void main( void )
{
    OSCCON = 0x72;            //8MHz clock
    while(!OSCCONbits.IOFS);  //Wait for OSC to become stable

  // configure A/D convertor
  OpenADC( ADC_FOSC_4      &
           ADC_LEFT_JUST   &
           ADC_2_TAD,
           ADC_CH0         &
           ADC_REF_VDD_VSS &
           ADC_INT_OFF, 1 );
     while(1){
        Delay10TCYx( 5 );     // Delay for 50TCY
        ConvertADC();         // Start conversion
```

Setting up the ADC and Collecting data from AN0 (Using ADC Header file) A bit more confusing … Part 2 of 2

```
        while( BusyADC() );   // Wait for completion
        result = ReadADC();   // Read result
    }
        CloseADC();           // Disable A/D converter
}
```

Using ADC - Manually

Setting up the ADC and Collecting data from AN0 (Manually)

```
#include <p18cxxx.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

#define ADCPin TRISAbits.TRISA0     // RA0/AN0 TRIS (DATA DIRECTION REGISTER)

unsigned int result;                //Result Variable

void main(void){
    OSCCON = 0x72;                  //8MHz clock
    while(!OSCCONbits.IOFS);        //Wait for OSC to become stable

    ADCPin = 1;                     //AN0 = Input
    ADCON0 = 0x00;                  //BIT 7:6 Always 00, BIT 5:2 is 0000 for Channel 0 (AN0),
                                    //BIT 1:0 is 00 (A/D Idle and Module is off) until we are done setting up
    ADCON1 = 0x0E;                  //BIT 7:6 Always 00, BIT 5 is 0 for VSS, BIT 4 is 0 for VDD
                                    //BIT 3:0 is 1110 for all digital expect AN0. We need it analog so we can convert it.
    ADCON2 = 0b00001100;            //BIT 7 is 0 for Left Justified, BIT 6 Always 0, BIT 5:3 001 for 2 TAD (A/D Acquisition Time)
                                    //BIT 2:0 is 100 for FOSC/4 for A/D Conversion Clock
    ADCON0bits.ADON = 1;            //Turn on the A/D module

    while(1){
        ADCON0bits.GO = 1;          //Start the conversion
        while(ADCON0bits.DONE);     //Wait until it's done
        result = ADRESH;            //Load ADRESH into result
        result <<= 8;               //Shift over Result 8 bits to the left
        result |= ADRESL;           //OR in our LOW byte to finish off out INT
        Delay10TCYx( 5 );           // Delay for 50TCY (precaution)
    }
}
```

Using PWM - <pwm.h>

Setting up the PWM for output.

```
#include <p18Cxxx.h>
#include <pwm.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

#define PWM_Tris TRISCbits.TRISC2                       //Define a nice name for the PWM TRIS bit.

char direction;                                         //direction holder for out loop
void main (void)
{
    char DutyCyc = 0x66;                                //Start Duty Cycle
    OSCCON = 0x72;                                      //8MHz clock
    while(!OSCCONbits.IOFS);                            //Wait for OSC to become stable

    PWM_Tris = 0;                                       //Set the pin a OUTPUT
    SetOutputPWM1 (SINGLE_OUT, PWM_MODE_1);             //Set as single output pwm and mode 1 (P1A,P1B,P1C,P1D active-high)
    OpenPWM1(DutyCyc);                                  //Set the duty cycle and start the PWM module

    while(1){
        if(DutyCyc == 0x12)                             //Check if we reached our minimum. If so set direction (up)
            direction = 1;
        if(DutyCyc == 0x66)                             //Check if we reached our maximum. If so clear direction (down)
            direction = 0;

        SetDCPWM1(DutyCyc);                             //Set the new duty cycle

        if(direction == 1)                              //based on direction if 1 (up) add 2 each time to the duty cycle
            DutyCyc+=2;
        else                                            // if 0 (down) minus 2 each time to the duty cycle
            DutyCyc-=2;
        Delay10KTCYx(3);                                //small delay (15mS)
    }
}
```

Using I2C – Bit Bang - X2402PI: 256x8 EEPROM

Bit Bang I2C … Part 1 of 3

```
#include <p18Cxxx.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

// DELAYS NOTE:
// 10TCY @ 8Mhz = 5uS
// Delay10TCYx(3); would then = 15uS

#define SDO LATCbits.LATC5              //Serial Data Out
#define SCL LATCbits.LATC3              //Serial Clock

#define SDI PORTCbits.RC5               //Used when SDO is input
#define SCK PORTCbits.RC3               //Used when SCL is input

#define SDOT TRISCbits.TRISC5           //SDO Tris Control
#define SCLT TRISCbits.TRISC3           //SCL Tris Control

void i2c_start(void);
void i2c_stop(void);
char i2c_ack(void);
void i2c_write(unsigned char byte);
unsigned char i2c_read(void);

void WriteX2402PI(unsigned char Address, unsigned char Data);
unsigned char ReadX2402PI(unsigned char Address);

unsigned char WAdd = 0xA0;      //Write address
unsigned char RAdd = 0xA1;      //Read  address

// Just some variables to hold our data
unsigned char temp;

void main (void)
{
    OSCCON = 0x72;                      //8MHz clock
    while(!OSCCONbits.IOFS);            //Wait for OSC to become stable

    ADCON1 = 0x0F;                      //Make all pins DIGITAL

    WriteX2402PI(0x00,0x4A);            //Write the letter J aka 0x4A to the device @ address 0x00
    temp = ReadX2402PI(0x00);           //Read from address 0x00 and temp should be the J aka 0x4A
    while(1);                           //Loop Forever
}
void i2c_start(void){
    SCLT = 0;                       //Clock = Output
    SCL = 0;                        //Clock = Low
    SDOT = 1;                       //Data = input(pulled high) so Data = 1
    Delay10TCY();                   //user may need to modify based on Fosc
    SCLT = 1;                       //Clock = input(pulled high) so Clock = 1
    Delay10TCY();                   //user may need to modify based on Fosc
    if(!SCK)                        //Check if the clock is low. If so then delay
        Delay10TCYx(3);             //user may need to modify based on Fosc
    SDOT = 0;                       //Data = Output
    SDO = 0;                        //Data = Low
    Delay10TCY();                   // user may need to modify based on Fosc
}

void i2c_stop(void){
    SCLT = 0;                       //Clock = Output
    SCL = 0;                        //Clock = Low
    SDOT = 0;                       //Data = Output
    SDO = 0;                        //Data = Low
    Delay10TCY();                   //user may need to modify based on Fosc
    SCLT = 1;                       //Clock = input(pulled high) so Clock = 1
    Delay10TCY();                   //user may need to modify based on Fosc
    if(!SCK)                        //Check if the clock is low. If so then delay
        Delay10TCYx(3);             //user may need to modify based on Fosc
    SDOT = 1;                       //Data = input(pulled high) so Data = 1
    Delay1TCY();                    //user may need to modify based on Fosc
    Delay1TCY();                    //user may need to modify based on Fosc
}

char i2c_ack(void){
    SCLT = 0;                       //Clock = Output
    SCL = 0;                        //Clock = Low
    SDOT = 1;                       //Data = input(pulled high) so Data = 1
    Delay10TCY();                   //user may need to modify based on Fosc
    SCLT = 1;                       //Clock = input(pulled high) so Clock = 1
    Delay1TCY();                    //1 cycle delay
    Delay1TCY();                    //1 cycle delay
    if(!SCK)                        //Check if the clock is low. If so then delay
        Delay10TCYx(3);             //user may need to modify based on Fosc

    if(SDI)                         //error if ack = 1, slave did not ack
```

Bit Bang I2C … Part 2 of 3

```
        return ( 1 );               //return with acknowledge error
    else
        return ( 0 );               // return with no error

}

void i2c_write(unsigned char byte){
    char x,tmp;
    for(x=0;x<8;x++){                   //Loop through 8 bits
        SCLT = 0;                       //Clock = Output
        SCL = 0;                        //Clock = Low

        if((byte & 0x80) != 0){         //BYTE AND 0x80 (0b10000000) (test if BIT7 != 0)
            SCLT = 0;                   //Clock = Output
            SCL = 0;                    //Clock = Low
            SDOT = 1;                   //Data = input(pulled high) so Data = 1
            Delay10TCY();               //user may need to modify based on Fosc
            SCLT = 1;                   //Clock = input(pulled high) so Clock = 1
            Delay10TCY();               //user may need to modify based on Fosc
            if(!SCK)                    //Check if the clock is low. If so then delay
                Delay10TCYx(3);         //user may need to modify based on Fosc
        } else {
            SCLT = 0;                   //Clock = Output
            SCL = 0;                    //Clock = Low
            SDOT = 0;                   //Data = Output
            SDO = 0;                    //Data = Low
            Delay10TCY();               //user may need to modify based on Fosc
            SCLT = 1;                   //Clock = input(pulled high) so Clock = 1
            Delay10TCY();               //user may need to modify based on Fosc
            if(!SCK)                    //Check if the clock is low. If so then delay
                Delay10TCYx(3);         //user may need to modify based on Fosc
        }

        byte <<= 1;                     //Shift Byte to LEFT by 1.
    }

}

unsigned char i2c_read(void){
    char x,tmp;
    SCLT = 0;                       //Clock = Output
    SCL = 0;                        //Clock = Low

    for(x=0;x<8;x++){               //Loop through 8 bits

    SCLT = 0;                       //Clock = Output
    SCL = 0;                        //Clock = Low
    SDOT = 1;                       //Data = input(pulled high) so Data = 1
    Delay10TCY();                   //user may need to modify based on Fosc
    SCLT = 1;                       //Clock = input(pulled high) so Clock = 1
    Delay1TCY();                    //user may need to modify based on Fosc
    Delay1TCY();                    //user may need to modify based on Fosc
    if(!SCK)                        //Check if the clock is low. If so then delay
         Delay10TCYx(3);            //user may need to modify based on Fosc

    tmp <<= 1;                      //Shift to LEFT by 1
    tmp &= 0xFE;                    //Make sure BIT 0 is a 0

    if(SDI)                         //If Data In is a 1 then OR it, if not then leave 0
        tmp |= 0x01;
    }

    return tmp;                     //Complete so return tmp (our received data)
}

void WriteX2402PI(unsigned char Address, unsigned char Data){
    char result;
    i2c_start();                    //Send Start

    i2c_write(WAdd);                //Send our Device Address with the Write BIT
    result = i2c_ack();             //Test for ACK

    i2c_write(Address);             //Send our Memory Address
    result = i2c_ack();             //Test for ACK

    i2c_write(Data);                //Send our Data
    result = i2c_ack();             //Test for ACK

    i2c_stop();                     //Send Stop
    Delay10KTCYx(2);                //Delay at least 10mS before anything else is done
}

unsigned char ReadX2402PI(unsigned char Address){
    unsigned char tmp;
    char result;

    i2c_start();                    //Send Start
```

Bit Bang I2C … Part 3 of 3

```
    i2c_write(WAdd);                //Send our Device Address with the Write BIT
    result = i2c_ack();             //Test for ACK

    i2c_write(Address);
    result = i2c_ack();             //Test for ACK

    i2c_start();                    //Send Start

    i2c_write(RAdd);                //Send our Device Address with the Read BIT
    result = i2c_ack();             //Test for ACK

    tmp = i2c_read();                 //Read 1 Byte from device and save in tmp

    i2c_stop();                     //Send Stop

    return tmp;
}
```

Using SPI – Bit Bang - DS1305: Serial Alarm Real-Time Clock

Using Bit Bang SPI and controlling a DS1305. Include BCD to DEC to BCD to ASCII conversions… Part 1 of 3

```
#include <p18Cxxx.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

#define CS  LATDbits.LATD0          //Chip Select Latch

#define SCL LATCbits.LATC3          //Clock Latch
#define SDO LATCbits.LATC5          //Data Out Latch
#define SDI PORTCbits.RC4           //Data In Port

#define SCLT TRISCbits.TRISC3       //Clock TRIS
#define SDOT TRISCbits.TRISC5       //Data Out TRIS
#define SDIT TRISCbits.TRISC4       //Data In TRIS
#define CST  TRISDbits.TRISD0       //Chip Select TRIS

void main(void);
void initSPI(void);
unsigned char ReadSPI(unsigned char address);
void WriteSPI(unsigned char address, unsigned char data);
unsigned char SPI_RByte(void);
void SPI_WByte(unsigned char data);
unsigned char BCDtoDEC(unsigned char BCD, char type);
unsigned char DECtoBCD(unsigned char DEC);
unsigned char DECtoASCII(unsigned char DEC,char part);

unsigned char Seconds;
unsigned char SecondBCD;
unsigned char SecH;
unsigned char SecL;

void main (void)
{
    unsigned char temp;


    OSCCON = 0x72;                  //8MHz clock
    while(!OSCCONbits.IOFS);        //Wait for OSC to become stable

    initSPI();                      //Initiate the lines for SPI

    WriteSPI(0x8F, 0x00);           //activate the OSC on IC

    while(1){
        temp = ReadSPI(0);              //Read register 0 aka seconds

        Seconds = BCDtoDEC(temp,'s');   //Convert The BCD to Decimal... now the user can just +1 or use as actual time to manipulate
        SecondBCD = DECtoBCD(Seconds);  //convert back to BCD to place back in the device

        SecH = DECtoASCII(Seconds,'h'); //Convert to ASCII so you can display on a LCD, GLCD, UART etc. TENS POSITION
        SecL = DECtoASCII(Seconds,'l'); //Convert to ASCII so you can display on a LCD, GLCD, UART etc. ONES POSITION

        Delay10KTCYx(100);              //500mS delay

    }
}
unsigned char DECtoASCII(unsigned char DEC,char part){
    unsigned char temp;
    unsigned char myData;

        temp = DECtoBCD(DEC);               //Convert the Decimal into BCD

        if(part == 'l')                     //If part = l (LOW PART)
            myData = temp & 0x0F;           //AND the byte with 0000-1111 to clear the high part of the byte
```

Using Bit Bang SPI and controlling a DS1305. Include BCD to DEC to BCD to ASCII conversions… Part 2 of 3

```
                                    //leaving the low part intact.
        if(part == 'h')             //If part = h (HIGH PART)
            myData = (temp >> 4) & 0x0F;    //Shift the byte to the right by 4 and then clear the high part (just incase)

        myData += 0x30;             //Add 0x30 to the value to make it ASCII

        return myData;              //Return myData
}
unsigned char DECtoBCD(unsigned char DEC){
    unsigned char temph;
    unsigned char templ;
    unsigned char myData;

        templ = DEC % 10;           //Get the remainder of the Decimal dived by 10 aka if DEC = 13 then 13 % 10 = 3
        temph = DEC / 10;           //Divide Decimal by 10 to get a count of 10's so if DEC = 13 then 13/10 = 1
        myData = (temph << 4) | templ;      //but them into 1 byte by shifting the high by to left by 4 and loading the low
                                    //part into the byte so if templ = 3 and temph = 1 you should get 0x13
        return myData;              //Return myData
}
unsigned char BCDtoDEC(unsigned char BCD, char type){
    unsigned char temp;
    unsigned char myData;
    char x;

    if((type == 's') ||(type == 'm')){          //Seconds or Minutes
        myData = BCD & 0x0F;                    //Clear the HIGH part of the byte and stay with the low part(ex: 0x23 would be 0x03)
        temp = (BCD >> 4) & 0b00000111;         //shift to the right 4 times to get high byte but we only need BITS 2:0 clear the rest (ex:0x23 would be 0x02)
        for(x=0;x<temp;x++){                    //Loop temp times so if temp was 2 then 2 time. temp will hold how many 10s there are
            myData += 10;                       //Add 10 every loop to the low part(ex: mydata holds 3 temp holds 2 so loop 2 times 10+10 = 20 + 3 = 23)
        }
        return myData;                          //return myData
    }
    if(type == 'h'){                            //Hour
        myData = BCD & 0x0F;                    //Same as above
        temp = (BCD >> 4) & 0b00000001;         //Same as above but we only need the first bit from high byte aka bit 4 (byte = BIT7:0)
        for(x=0;x<temp;x++){                    //same as above
            myData += 10;                       //same as above
        }
        return myData;                          //return myData
    }
    if(type == 'p'){                            //AM/PM
        myData = (BCD >> 5) & 0b00000001;       //Same as above but we shift over 5 times since we need and bit of high part aka bti 5 (byte = BIT7:0)
        return myData;                          //return myData
    }
    if(type == 'd'){                            //DAY
        myData = BCD & 0b00001111;              //For the day we need only the LOW part of byte so we clear the high part (IF ANY)
        return myData;                          //return myData
    }
    if((type == 'D') ||(type == 'M')){          //DATE or MONTH
        myData = BCD & 0x0F;                    //same as above
        temp = (BCD >> 4) & 0b00000011;         //same as above but we only use the first 2 bits of high byte
        for(x=0;x<temp;x++){                    //same as above
            myData += 10;                       //same as above
        }
        return myData;                          //return myData
    }
    if(type == 'Y'){                            //DATE or MONTH
        myData = BCD & 0x0F;                    //same as above
        temp = (BCD >> 4) & 0b00001111;         //same as above but we use the whole upper part
        for(x=0;x<temp;x++){                    //same as above
            myData += 10;                       //same as above
        }
        return myData;                          //return myData
    }
}
void initSPI(void){
    SCLT = 0;           //Set the SCL (CLOCK)       pin to output
    SDOT = 0;           //Set the SDO (DATA OUT)    pin to output
    SDIT = 1;           //Set the SDI (DATA IN)     pin to input
    CST = 0;            //Set the CS  (Chip Select) pin to output

    CS = 0;             //Set Chip Select Low
    SDO = 0;            //Set Serial Data Out Low
    SCL = 0;            //Set Serial Clock Low
}
void SPI_WByte(unsigned char data){
    char x;
    for(x=0;x<8;x++){               //Loop 8 bits
        SCL = 0;                    //Clock Low
        SDO = 0;                    //Data Low
        if((data & 0x80) != 0)      //Clear entire byte except bit 7 and check if byte is greater than 0
            SDO = 1;                //Set the Data Out to HIGH
        Delay10TCY();               //Delay 5uS
        SCL = 1;                    //Clock High
```

Using Bit Bang SPI and controlling a DS1305. Include BCD to DEC to BCD to ASCII conversions… Part 3 of 3

```
        Delay10TCY();               //Delay 5uS

        data <<= 1;                 //Shift data bit out to the right
    }
}
unsigned char SPI_RByte(void){
    char x,data;
    for(x=0;x<8;x++){           //Loop 8 bits
        SCL = 0;                //Clock Low
        Delay10TCY();           //5uS delay

        data <<= 1;             //Shift 1 bit to the left
        data &= 0xFE;           //Clear BIT 0 from byte without touching the rest

        if(SDI)                 //if the Serial Data In is HIGH do below if not then leave it 0
            data |= 1;          //OR in a 1 into out byte

        Delay10TCY();           //delay 5uS
        SCL = 1;                //Clock High

    }
    return data;                //return our data
}
void WriteSPI(unsigned char address, unsigned char data){
    SCL = 1;                //Clock High
    CS = 1;                 //Chip Select High
    SPI_WByte(address);     //Write our address
    SPI_WByte(data);        //write our data
    CS = 0;                 //Chip Select Low
}
unsigned char ReadSPI(unsigned char address){
    unsigned char tmp;

    SCL = 1;                //Clock High
    CS = 1;                 //Chip Select High
    SPI_WByte(address);     //Write our address to read from
    tmp = SPI_RByte();      //Get the byte from the device
    CS = 0;                 //Chip Select Low

    return tmp;             //Return tmp
}
```

LCD Control (2x16 – HD44780)

Setting up the LCD and Sending Strings… Part 1 of 2

```
#include <p18Cxxx.h>
#include <delays.h>
#pragma config WDT = OFF, LVP = OFF, OSC = INTIO67, XINST = OFF

void SendLCD(unsigned char Byte, char type);
void E_TOG(void);
void SendNyb(unsigned char Byte);
void LCD_Init(void);
void SetLine (char line);
void LCD_String(unsigned rom char *string);
/*
PINOUT:
LCD   -   PIC
-------------
DB4   -   RA0
DB5   -   RA1
DB6   -   RA2
DB7   -   RA3
RS    -   RA4
E     -   RA5
R/W   -   GND
*/
#define LCD LATA                    //LCD Latch PORT
#define LCD_RS LATAbits.LATA4       //Register Select (0: Instruction/ 1: Data)
#define LCD_E LATAbits.LATA5        //Enable (Starts data read/write)
#define LCDT TRISA                  //LCD Tris port

void main (void)
{
    char x,tmp;                 //Some Variables
    OSCCON = 0x72;              //8MHz clock
    while(!OSCCONbits.IOFS);    //Wait for OSC to become stable

    ADCON1 = 0x0F;              //Make all pins digital
    LCD_Init();                 //Initialize the LCD

    SetLine(1);                 //Set Line 1
    LCD_String((unsigned rom char*)"  AtomSoftTech  "); //Send out string
    SetLine(2);                 //Set Line 2
    LCD_String((unsigned rom char*)" 2x16 LCD Demo. "); //Send out string

    while(1){

    }
}
void LCD_String(unsigned rom char *string){
    while(*string != 0){            //While the data pointed to isnt a 0x00 do below
        SendLCD(*string++,1);       //Send out the current byte pointed to
    }
}
void LCD_Init(void){
    LCDT = 0x00;        //Set LCD Pins as output
    LCD &= 0xF0;        //clear only te db4:db7 pins

    Delay10KTCYx(3);    //delay 15mS
    SendNyb(0x03);      //Function set (Interface is 8 bits long.)
    Delay10KTCYx(1);    //delay 5mS
    SendNyb(0x03);      //Function set (Interface is 8 bits long.)
    Delay100TCYx(3);    //delay 150uS
    SendNyb(0x03);      //Function set (Interface is 8 bits long.)
    Delay100TCYx(3);    //delay 150uS
    SendNyb(0x02);      //Function set (Interface is 8 bits long.)
    Delay100TCYx(1);    //delay 50uS
    SendLCD(0x28,0);    //Function set (Interface is 4 bits long.) 2-lines, 5x8 dots
    Delay100TCYx(1);    //delay 50uS
    SendLCD(0x10,0);    //Cursor move, Shift to the Left
    Delay100TCYx(1);    //delay 50uS
    SendLCD(0x06,0);    //Increment
    Delay100TCYx(1);    //delay 50uS
    SendLCD(0x0D,0);    //Sets entire display On ,cursor Off, and blinking of cursor position ON
    Delay100TCYx(1);    //delay 50uS
    SendLCD(0x01,0);    //Clears entire display and sets DDRAM address 0 in address counter.
    Delay10KTCYx(1);    //delay 5mS
}
void E_TOG(void){
    LCD_E = 1;          //Set Enable High
    Delay10TCY();       //Delay 5uS
    LCD_E = 0;          //Set Enable Low
}
void SetLine (char line){
    if(line == 1) SendLCD(0x80,0);      //Send 0x80 to set line to 1
    if(line == 2) SendLCD(0xC0,0);      //Send 0xC0 to set line to 2
    Delay100TCYx(1);                    //delay 50uS
}
```

Setting up the LCD and Sending Strings… Part 2 of 2

```
void SendLCD(unsigned char Byte, char type){
    char ByteL;
    LCD_RS = type;          //Set whether it is a Command (0) or Data/Char (1)
    ByteL = Byte & 0x0F;    //Place Byte in ByteL and Clear the high part of byte
    Byte >>= 4;             //Shift over Byte by 4

    LCD &= 0xF0;            //Clear the LCD portion on the PORTA only
    LCD |= Byte;            //OR the new data to the LCD portion of PORTA (sending high part of byte)
    E_TOG();                //Toggle the E(enable)
    Delay10TCY();           //Delay 5uS
    LCD &= 0xF0;            //Same as above
    LCD |= ByteL;           //Same as above (sending low part of byte)
    E_TOG();                //same as above
}
void SendNyb(unsigned char Byte){
    Byte &=0x0F;        //Clear the top half of byte
    LCD &= 0xF0;        //Clear the LCD half of PORTA
    LCD |= Byte;        //OR the new data into LCD part of PORTA
    E_TOG();            //Toggle E(enable)
}
```