```
/* Microchip Technology Inc. and its subsidiaries.  You may use this software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
 * TERMS.
 */

/*
 * File: RTC.h
 * Author: velmurugan S
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
// Microchip Technology Inc. and its subsidiaries provide a usage disclaimer

// Include guard to prevent double inclusion of the header file
#ifndef RTC_H
#defineRTC_H

// Include a header file for Microchip PIC microcontrollers
#include <xc.h>

// Function prototype for setting time and calendar
void Set_Time_Calender(unsigned char, unsigned char, unsigned char, unsigned char, unsigned char, unsigned char, uns

// Function prototype for converting a value to BCD (Binary Coded Decimal)
unsigned char Bcd_Conversion(unsigned char);

// Function prototype for setting seconds
unsigned char Set_Seconds(unsigned char);

// Function prototype for setting minutes
unsigned char Set_Minute(unsigned char);

// Function prototype for setting standard hours
unsigned char Set_Standard_Hour(unsigned char, unsigned char);

// Function prototype for setting railway hours
unsigned char Set_Railway_Hour(unsigned char);

// Function prototype for setting the day
unsigned char Set_Day(unsigned char);

// Function prototype for setting the date
unsigned char Set_Date(unsigned char);

// Function prototype for setting the month
unsigned char Set_Month(unsigned char);

// Function prototype for setting the year
unsigned char Set_Year(unsigned char);
```

```
igned char, unsigned char);
```

```
// Function prototype for converting hours to a specific format
unsigned char Hour_Conversion(unsigned char);

// Function prototype for determining whether it's AM or PM
unsigned char Am_or_Pm(unsigned char);

// Function prototype for reading the current time
void Read_Time(void);

// Function prototype for reading the current date
void Read_Date(void);

// Function prototype for converting the least significant BCD digit to a character
unsigned char Bcd_to_Lsb(unsigned char);

// Function prototype for converting the most significant BCD digit to a character
unsigned char Bcd_to_Msb(unsigned char);

// Function prototype for extracting the most significant bits of the hour
unsigned char Hour_Msb(unsigned char);

// Declaration of a temporary variable
unsigned char temp;

// Array to store the current time as characters
unsigned char Current_Time[11] = "  -  -    ";

// Array to store the current date as characters
unsigned char Current_Date[9] = "  -  -  ";



void Read_Time(){

    unsigned char c;

    temp=I2C_Read_Data(1);
        Current_Time[7]=Bcd_to_Lsb(temp);
        Current_Time[6]=Bcd_to_Msb(temp);
  temp=I2C_Read_Data(1);
        Current_Time[4]=Bcd_to_Lsb(temp);
        Current_Time[3]=Bcd_to_Msb(temp);
 temp=I2C_Read_Data(1);
        Current_Time[1]=Bcd_to_Lsb(temp);
        Current_Time[0]=Hour_Msb(temp);

        if((c=Am_or_Pm(temp))){
                Current_Time[9]=c;
                Current_Time[10]='M';

        }else{
        Current_Time[9]=c;
        }



}


void Read_Date(){

        I2C_Read_Data(1);


        temp=I2C_Read_Data(1);
```

```
        Current_Date[0]=Bcd_to_Msb(temp);
        Current_Date[1]=Bcd_to_Lsb(temp);

        temp=I2C_Read_Data(1);


        Current_Date[3]=Bcd_to_Msb(temp);
        Current_Date[4]=Bcd_to_Lsb(temp);

        temp=I2C_Read_Data(0);


        Current_Date[6]=Bcd_to_Msb(temp);
        Current_Date[7]=Bcd_to_Lsb(temp);
}



void Set_Time_Calender(unsigned char Sec, unsigned char Min, unsigned char Mode,unsigned char Hour,unsigned char Day

    I2C_Device_Select_W((unsigned char)0xD0 ,(unsigned char) 0x00);

    I2C_Data_Write(Set_Seconds(Sec));

    I2C_Data_Write(Set_Minute(Min));

    if(Mode){
    I2C_Data_Write(Set_Standard_Hour(Hour,Mode));
    }
    else{
    I2C_Data_Write(Set_Railway_Hour(Hour));
    }

    I2C_Data_Write(Set_Day(Day));

    I2C_Data_Write(Set_Date(Date));

    I2C_Data_Write(Set_Month(Month));

    I2C_Data_Write(Set_Year(Year));

    }



unsigned char Set_Date(unsigned char Date){

    return Bcd_Conversion(Date);
}

unsigned char Set_Month(unsigned char Month){
    return Bcd_Conversion(Month);

 }

unsigned char Set_Year(unsigned char Year){
    return Bcd_Conversion(Year);

}

unsigned char Set_Standard_Hour(unsigned char hrs,unsigned char C){

 unsigned char temp;

temp=(C=='P')?0x60:0x40;
```

```
,unsigned char Month,unsigned char Year,unsigned char Date){
```

```
return Hour_Conversion(hrs)|temp;


}
unsigned char Set_Railway_Hour(unsigned char hrs){

return (Hour_Conversion(hrs)&0x3F);

}

unsigned char Set_Minute(unsigned char min){

 return Bcd_Conversion(min);

}


unsigned char Set_Day(unsigned char day){

    return day;

}

unsigned char Set_Seconds(unsigned char seconds){

 return Bcd_Conversion(seconds);


}

unsigned char Bcd_Conversion(unsigned char value){
    return (unsigned char)(((value/10)<<4)|(value%10));
}

unsigned char Hour_Conversion(unsigned char hrs){

if( hrs >= 20){
    return (unsigned char) (1<<5 | hrs%10);
}
else if(hrs >= 10)
 return (unsigned char)((1<<4 | hrs%10));

 return hrs;
}

unsigned char Hour_Msb(unsigned char hrs){

    if(hrs&0x40){
   if(hrs&0x40){
       if(hrs&0x10){
           return '1';
       }
       return '0';
   }else{
       if(hrs&0x20)
           return '2';
       else if(hrs&0x10)
           return '1';
       else
           return '0';

   }


/*
   if((hrs&0x40) && (hrs&0x10)){
       return '1';
   }else{
```

```
   if( (!(hrs&0x04))&& hrs&0x20) {
     return '2';
   }
   else if(hrs&0x10){
      return '1';
   }else{}

    }
    return '0';
*/
}



 unsigned char Am_or_Pm(unsigned char hrs){

     if(hrs&0x40){
         if(hrs&0x20){
             return 'P';
         }
         return 'A';
     }else{
         return 0;
     }
 }

unsigned char Bcd_to_Lsb(unsigned char value){


    return (value&0x0F)+48;
}


unsigned char Bcd_to_Msb(unsigned char value){

    return (value>>4)+48;
}
#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

   void Set_Time_Calender(unsigned char , unsigned char , unsigned char ,unsigned char ,unsigned char,unsigned char

unsigned char Bcd_Conversion(unsigned char );

unsigned char Set_Seconds(unsigned char);

unsigned char Set_Minute(unsigned char );

unsigned char Set_Standard_Hour(unsigned char , unsigned char );

unsigned char Set_Railway_Hour(unsigned char );

unsigned char Set_Day(unsigned char );

unsigned char Set_Date(unsigned char );

unsigned char Set_Month(unsigned char);

unsigned char Set_Year(unsigned char );

unsigned char Hour_Conversion(unsigned char );

unsigned char Am_or_Pm(unsigned char );

void Read_Time(void);

void Read_Date(void);
```

```
,unsigned char,unsigned char );
```

```
unsigned char Bcd_to_Lsb(unsigned char );

unsigned char Bcd_to_Msb(unsigned char );

unsigned char Hour_Msb(unsigned char );


#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif/* XC_HEADER_TEMPLATE_H */
```