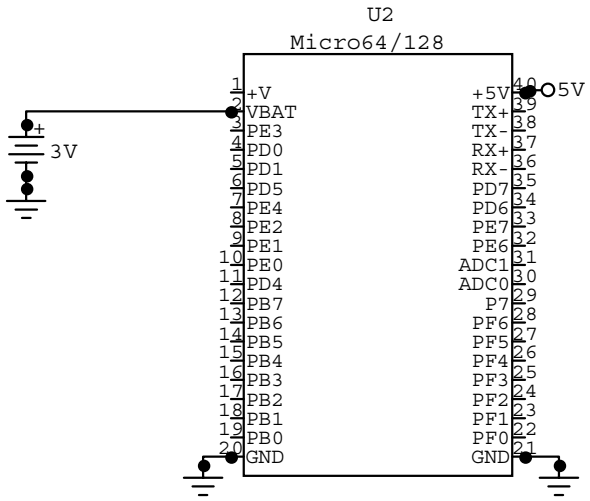
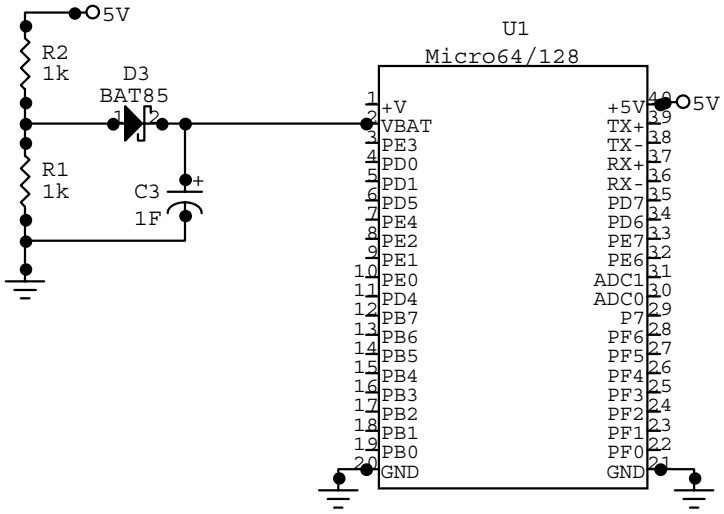
Micromint, Inc	**AN704**
	Micro64/128
I^2C Real Time Clock Calendar	12/3/04

Introduction: This application note demonstrates how to read and write the to the I^2C Real Time Clock Calendar. It also demonstrates how to use a Coin Battery or a Super Capacitor for battery backup.

Background: Micro64/128 has an I^2C Real Time Clock (RTC) built into it. There are two ways that the clock can be backed up so that it continues to increment if power is lost, by battery or by super capacitor. The battery is pretty straight forward. All you have to do is connect a 3V battery to VBAT. The following schematic demonstrates connecting a battery to the Micro64/128.



The following schematic demonstrates connecting a super capacitor to VBAT:



How it works: The RTC uses the I^2C bus to communicate to the mega64 or mega128 controller. The Micro64/128 has utilities for the following functions for the RTC:

Read & Write Functions	Write Only Functions
Tenth of a Second Register	Enable The Alarm
Seconds Register	Disable the Alarm
Minutes Register	Set Alarm to Repeat every Second
Hours Register	Set Alarm to Repeat every Minute
Day of the Week Register	Set Alarm to Repeat every Hour
Day of the Month Register	Set Alarm to Repeat every Day
Month	Set Alarm to Repeat every Month
Year Register	Set Alarm to Repeat every Year
Alarm Seconds Register	**Read Only Functions**
Alarm Minutes Register	Read the Alarm Flags
Alarm Hours Register	
Alarm Day of the Month Register	

The RTC's interrupt pin is connected to PORTE bit 5(PE5). The CodeVision AVR program demonstrates how to use the Micro64/128's utilities to access the RTC.

Program Listing:

```
/*****************************************
Program : RTC Example for Micro64
Company : Micromint, Inc
*****************************************/

#include <mega64.h>
#include <MMRS485.h>     // Micromints Library for using both USARTs
#include <stdio.h>       // Standard I/O library
#include <delay.h>       // Library for delays
#include <bcd.h>
#include <stdlib.h>

unsigned int Pass @0xFFE;
unsigned char CLDATA @ 0xFFD;

#define MAX_ENTRY_LENGTH 10
void(*RDTENTHSECOND)(void)=0x7cf3;
void(*RDSEC)(void)=0x7cf7;
void(*RDMIN)(void)=0x7cfc;
void(*RDHOUR)(void)=0x7d00;
void(*RDDOW)(void)=0x7d05;
void(*RDDOM)(void)=0x7d09;
void(*RDMONTH)(void)=0x7d0d;
void(*RDYEAR)(void)=0x7d11;

void(*WRSEC)(void)=0x7d15;
void(*WRMIN)(void)=0x7d1D;
void(*WRHOUR)(void)=0x7d21;
void(*WRDOW)(void)=0x7d29;
void(*WRDOM)(void)=0x7d2D;
void(*WRMONTH)(void)=0x7d31;
void(*WRYEAR)(void)=0x7d35;

void(*RDALSEC)(void)=0x7d39;
void(*RDALMIN)(void)=0x7d3E;
void(*RDALHOUR)(void)=0x7d43;
void(*RDALDOM)(void)=0x7d48;
void(*RDALMONTH)(void)=0x7d4D;

void(*WRALSEC)(void)=0x7d58;
void(*WRALMIN)(void)=0x7D6B;
void(*WRALHOUR)(void)=0x7D7E;
void(*WRALDOM)(void)=0x7D91;
void(*WRALMONTH)(void)=0x7DA4;
```

```
void(*ENABLEALARM)(void)=0x7DB9;
void(*DISABLEALARM)(void)=0x7DC7;

void(*ALARMREPEATSEC)(void)=0x7DD2;
void(*ALARMREPEATMIN)(void)=0x7DFB;
void(*ALARMREPEATHOUR)(void)=0x7E24;
void(*ALARMREPEATDAY)(void)=0x7E4D;
void(*ALARMREPEATMONTH)(void)=0x7E76;
void(*ALARMREPEATYEAR)(void)=0x7EA2;


void(*RDALARMFLAGS)(void)=0x7ECB;


void disMainMenu(void);
void setClock(void);
void setAlarm(void);
void readAlarm(void);
void enable_Alarm(void);
void disable_Alarm(void);
void setAlarmRepeat(void);
int getBCD(void);
unsigned char checkUSART1(void);
// External Interrupt 5 service routine
interrupt [EXT_INT5] void ext_int5_isr(void)
{
#asm("cli")
(*RDALARMFLAGS)();
printf("Alarm Flags = %02d\r\n", Pass);
printf("\r\n");
printf("\r\n");
printf("\r\n");
printf("An Alarm Occured\r\n");
printf("\r\n");
printf("\r\n");
printf("\r\n");
#asm("sei")
}

// Declare your global variables here
int COM;        // if COM = 0 then use USART0 if it = 1 then use USART1

void main(void)
{
// Declare your local variables here

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: On
// INT5 Mode: Low level
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x20;
EIFR=0x20;

// Set up USART1's Baud rate at 9600 bps with a 11.0592 MHz Crystal
UCSR1A=0x00;      // RX EN, TX EN
UCSR1B=0x18;     // RX EN, TX EN
UCSR1C=0x06;     // 8N1
UBRR1H=0x00;    // Baud rate high - 9600
UBRR1L=0x47;   // Baud rate low

delay_ms(1000);
   COM = 1;                     // Use USART1
   DDRD.6 = 0;                                      // Make PORTD.6 an output
   PORTD.6 = 1;                                     // Enable the RS485 control line
   printf("Press any key to display the main menu\r\n");
   printf("\r\n");
while (1)
```

```
    {
     (*RDDOW)();
    switch(Pass)
     {
      case 1 :
        printf("Sunday ");
        break;
      case 2 :
        printf("Monday ");
        break;
      case 3 :
        printf("Tuesday ");
        break;
      case 4 :
        printf("Wednesday ");
        break;
      case 5 :
        printf("Thursday ");
        break;
      case 6 :
        printf("Friday ");
        break;
      case 7 :
        printf("Saturday ");
        break;
     }

    (*RDMONTH)();
    printf("%02d",bcd2bin(Pass));
    (*RDDOM)();
    printf("/%02d",bcd2bin(Pass));
    (*RDYEAR)();
    printf("/%02d",bcd2bin(Pass));
    (*RDHOUR)();
    printf(" %02d",bcd2bin(Pass));
    (*RDMIN)();
    printf(":%02d",bcd2bin(Pass));
    (*RDSEC)();
    printf(":%02d",bcd2bin(Pass));
    (*RDTENTHSECOND)();
    printf(":%02d",bcd2bin(Pass));
    printf("\r");
    if (checkUSART1()>0)
      disMainMenu();
    };
}

void disMainMenu(void)
{
char rec;
   printf("\r\n");
   printf("Main Menu \r\n");
   printf("1 - Set the Clock\r\n");
   printf("2 - Set the Alarm\r\n");
   printf("3 - Read the Clock\r\n");
   printf("4 - Read the Alarm\r\n");
   printf("5 - Enable the Alarm\r\n");
   printf("6 - Disable the Alarm\r\n");
   printf("7 - Set how often the Alarm Repeats\r\n");
   rec = getchar();
      switch(rec)
      {
       case '1' :
          setClock();
          break;
       case '2' :
          setAlarm();
          break;
       case '3' :
          break;
       case '4' :
          readAlarm();
          break;
       case '5' :
          enable_Alarm();
```

```
            break;
          case '6' :
            disable_Alarm();
            break;
          case '7' :
            setAlarmRepeat();
            break;
        }
        printf("Press any key to display the main menu\r\n");
        printf("\r\n");
}

void setClock(void)
{
unsigned char Seconds;
unsigned char Minutes;
unsigned char Hours;
unsigned char Month;
unsigned char DOM;
unsigned char DOW;
   printf("Please enter what you would like to set the seconds to.\r\n");
   Seconds = getBCD();
   printf("Please enter what you would like to set the minute to.\r\n");
   Minutes = getBCD();
   printf("Please enter what you would like to set the hour to.\r\n");
   Hours = getBCD();
   printf("Please enter what you would like to set the Day of the Week to.\r\n");
   printf("1 - Sunday\r\n");
   printf("2 - Monday\r\n");
   printf("3 - Tuesday\r\n");
   printf("4 - Wednesday\r\n");
   printf("5 - Thursday\r\n");
   printf("6 - Friday\r\n");
   printf("7 - Saturday\r\n");
   DOW = getBCD();
   printf("Please enter what you would like to set the Month to.\r\n");
   Month = getBCD();
   printf("Please enter what you would like to set the Day of the Month to.\r\n");
   DOM = getBCD();
   printf("Please enter what you would like to set the Year to.\r\n");
   CLDATA = getBCD();
   (*WRYEAR)();
   CLDATA = Seconds;
   (*WRSEC)();
   CLDATA = Minutes;
   (*WRMIN)();
   CLDATA = Hours;
   (*WRHOUR)();
   CLDATA = DOW;
   (*WRDOW)();
   CLDATA = Month;
   (*WRMONTH)();
   CLDATA = DOM;
   (*WRDOM)();
}
void setAlarm(void)
{
   printf("Please enter what you would like to set the Alarm seconds to.\r\n");
   CLDATA = getBCD();
   (*WRALSEC)();
   printf("Please enter what you would like to set the Alarm minutes to.\r\n");
   CLDATA = getBCD();
   (*WRALMIN)();
   printf("Please enter what you would like to set the Alarm hour to.\r\n");
   CLDATA = getBCD();
   (*WRALHOUR)();
   printf("Please enter what you would like to set the Alarm Month to.\r\n");
   CLDATA = getBCD();
   (*WRALMONTH)();
   printf("Please enter what you would like to set the Alarm Day of the Month to.\r\n");
   CLDATA = getBCD();
   (*WRALDOM)();
}
void readAlarm(void)
{
```

```
    printf("\r\n");
    printf("The alarm is set to ");
    (*RDALMONTH)();
    printf("%02d",bcd2bin(Pass));
    (*RDALDOM)();
    printf("/%02d",bcd2bin(Pass));
    (*RDALHOUR)();
    printf(" %02d",bcd2bin(Pass));
    (*RDALMIN)();
    printf(":%02d",bcd2bin(Pass));
    (*RDALSEC)();
    printf(":%02d",bcd2bin(Pass));
    printf("\r\n");
    printf("\r\n");
}
void enable_Alarm(void)
{
// Global interrupts enable
#asm("sei")
(*ENABLEALARM)();
}

void disable_Alarm(void)
{
// Global interrupts disabled
#asm("cli")
(*DISABLEALARM)();
}
void setAlarmRepeat(void)
{
char recx;
  printf("Please enter what you would like to set the alarm repeat to.\r\n");
  printf("1 - Once a Second\r\n");
  printf("2 - Once a Minute\r\n");
  printf("3 - Once a Hour\r\n");
  printf("4 - Once a Day\r\n");
  printf("5 - Once a Month\r\n");
  printf("6 - Once a Year\r\n");
  recx = getchar();
      switch(recx)
      {
       case '1' :
         (*ALARMREPEATSEC)();
         break;
       case '2' :
         (*ALARMREPEATMIN)();
         break;
       case '3' :
         (*ALARMREPEATHOUR)();
         break;
       case '4' :
         (*ALARMREPEATDAY)();
         break;
       case '5' :
         (*ALARMREPEATMONTH)();
         break;
       case '6' :
         (*ALARMREPEATYEAR)();
         break;
      }
}

int getBCD(void)
{
char mystr[MAX_ENTRY_LENGTH + 1]; //Declare a string
char c;

      COM = 1;
      DDRD.6 = 0;                           // Make PORTD.6 an output
      PORTD.6 = 1;                          // Enable the RS485 control line
      c = 0;
      while( c < MAX_ENTRY_LENGTH)
      {
       mystr[c++] = getchar();
       if (mystr[c-1] == 13)
```

```
            break;
        }
        mystr[c] = '\0';

        return  bin2bcd(atoi(mystr));
}
unsigned char checkUSART1(void)
{
unsigned char status;
status = UCSR1A;
status = status & 0x80; // mask out all of the bits in the UCSR1A register except the receive complete
switch(status)
        {
         case 0x0:      //No characters came into the USART
            return 0;
            break;
         case 0x80:    // A character came into the USART
            status = UDR1;
            return 255;
            break;
        }
}
```