LAB 7 SPI, I2C, and RTC

**OBJECTIVES**

1. Learn to use SPI interface
2. Learn to how to use SD-CARD
3. Understand SPI Interface
4. Learn to use I2 C interface
5. Understand I2 C Interface
6. Learn to how to use RTC

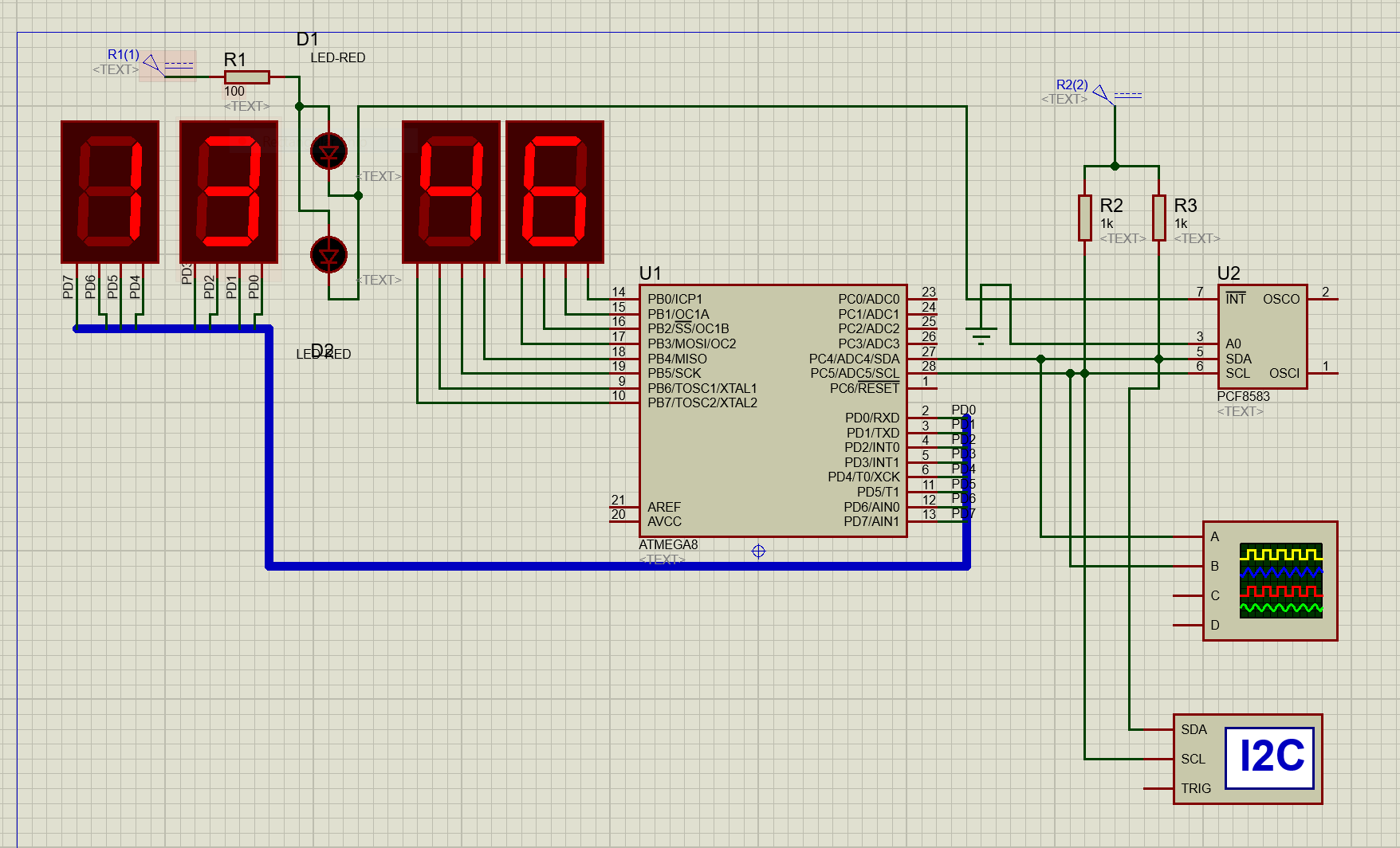
**Device:**

1. ATMEGA8
2. SD-CARD
3. PCF8583 RTC
4. Nokia LCD

**STEPS:**

**Part 1**: I2C with RTC

1. Study PCF8583 RTC IC on how to send/receive data from ATMega8 via I2C Interface
2. Using Proteus to create a circuit below:



1. Study code provided

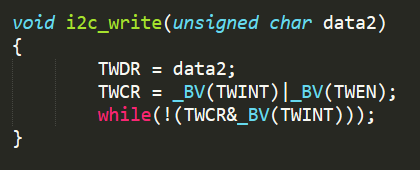
|  |  |
| --- | --- |
| 1 | #include <avr/io.h> |
| 2 | #include <avr/delay.h> |
| 3 | #define F\_CPU 8000000UL |
| 4 |  |
| 5 | *void* i2c\_write(*unsigned* *char* *data2*) { |
| 6 |  |
| 7 | TWDR = data2; |
| 8 | TWCR = \_BV(TWINT)|\_BV(TWEN); |
| 9 | while(!(TWCR&\_BV(TWINT))); |
| 10 | } |
| 11 |  |
| 12 | *unsigned* *int* i2c\_readwithoutack(*void*) { |
| 13 |  |
| 14 | TWCR = \_BV(TWINT) | \_BV(TWEN); |
| 15 | while(!(TWCR&\_BV(TWINT))); |
| 16 | return TWDR; |
| 17 | } |
| 18 | *unsigned* *int* i2c\_readwithack(*void*) { |
| 19 |  |
| 20 | TWCR = \_BV(TWINT) | \_BV(TWEN)|\_BV(TWEA); |
| 21 | while(!(TWCR&\_BV(TWINT))); |
| 22 | return TWDR; |
| 23 | } |
| 24 |  |
| 25 | *void* i2c\_start(*void*) { |
| 26 |  |
| 27 | TWCR = \_BV(TWINT) | \_BV(TWSTA)|\_BV(TWEN); |
| 28 | while((TWCR &\_BV(TWINT))==0); |
| 29 | } |
| 30 |  |
| 31 | *void* i2c\_stop(*void*) { |
| 32 |  |
| 33 | TWCR = \_BV(TWINT)|\_BV(TWEN)|\_BV(TWSTO); |
| 34 | } |
| 35 |  |
| 36 | *void* i2c\_init(*void*) { |
| 37 |  |
| 38 | TWSR = 0x00; |
| 39 | TWBR = 0x47; |
| 40 | TWCR = 0x04; |
| 41 | } |
| 42 |  |
| 43 | *int* main(*void*) { |
| 44 |  |
| 45 | DDRB = 0xff; |
| 46 | DDRD = 0xff; |
| 47 | i2c\_init();              // Set Clock control Register3 |
| 48 |  |
| 49 | i2c\_start(); |
| 50 | i2c\_write(0xA0); |
| 51 | i2c\_write(0x00); |
| 52 | i2c\_write(0x00); |
| 53 | i2c\_stop(); |
| 54 |  |
| 55 |  |
| 56 | i2c\_start(); |
| 57 | i2c\_write(0xA0); |
| 58 | i2c\_write(0x04); |
| 59 | i2c\_write(0x13); |
| 60 | i2c\_stop(); |
| 61 |  |
| 62 | \_delay\_ms(100); |
| 63 | while(1) { |
| 64 |  |
| 65 | i2c\_start(); |
| 66 | i2c\_write(0xA0); |
| 67 | i2c\_write(0x03); |
| 68 | i2c\_start(); |
| 69 | i2c\_write(0xA1); |
| 70 | PORTB=i2c\_readwithack();           // read minute |
| 71 | PORTD= (0x3f&i2c\_readwithoutack());         // read hour |
| 72 | i2c\_stop(); |
| 73 |  |
| 74 | } |
| 75 |  |
| 76 | } |

**Result:**

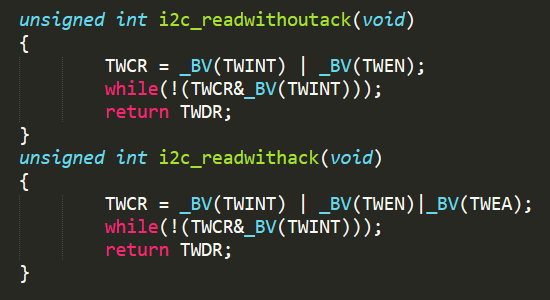
From the result, we see that when we start the simulation, it’ll have a small windows pop-up that is a current date and time of the computer. And set of 7-segments display time in hour and minute.

**Discussion:**

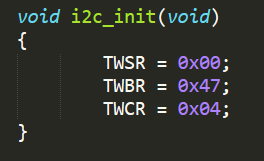
For all the function, we have to define each function before we use it, and we start with function as shown below:



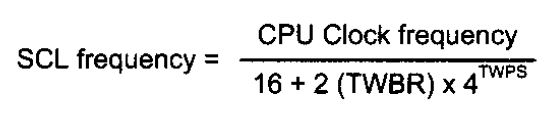
This function is a write function that we receive the data from main and set TWDR (TWI Data Register) with value of data2 and set TWINT and TWEN (set interrupt and Enable)



In PCF8583, there are required to have a 2 different read, So that we need to create 2 different functions. The first one is a read without ACK and another is read with ACK



This is an initial function. First, set TWSR to set prescaler bits to zero. Next, set SCL frequency from formula:

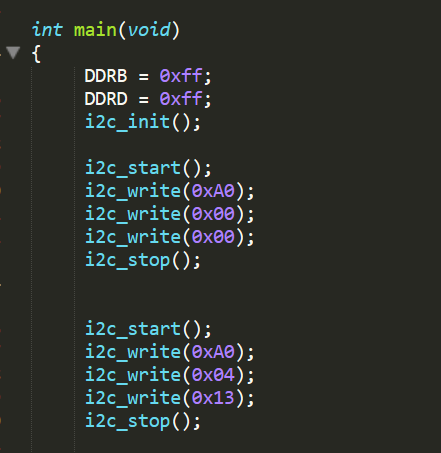


**When** TWBR = 0x478 = 7110

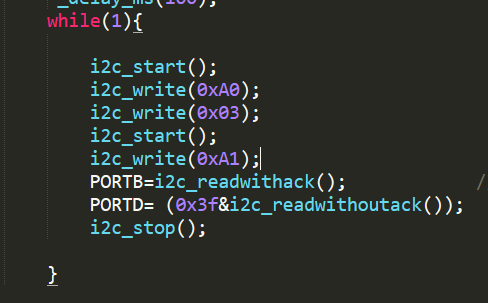
CPU Clock frequency (XTAL) = 8MHz

TWPS = 0

The SCL frequency is 50K Hz., and TWCR is an enable TWI module.

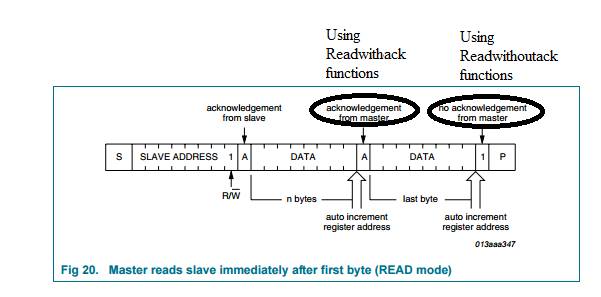


In main function, first we need reset the value and then we start again and select slave address (0xA0) then send the data (0x04), it means we will go to register section in slave (PCF8583), to select the hour section then we send data again (0x13) to set hour to be “13” in the RTC module and that’s why when we start simulation the hour part of display always be “13”.



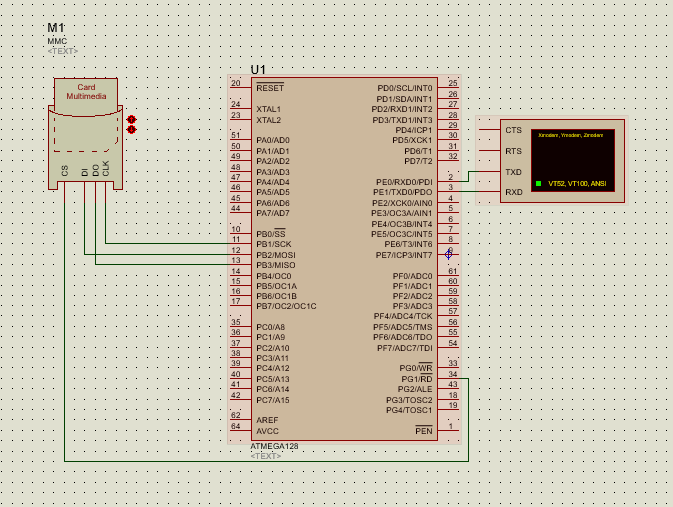
This part is similar previous part, we select a slave (0xA0) to send data then we select minute section in slave’s register. After that we select a slave (0xA1) to receive data from slave and we keep minute into port B and hour into port D.

And the reason we have to do (0x03f & read\_data) because first 2 bit(from MSB) is a setting format of time and we don’t need it for displaying on 7-seg



**Part 2**: SPI with SD-CARD

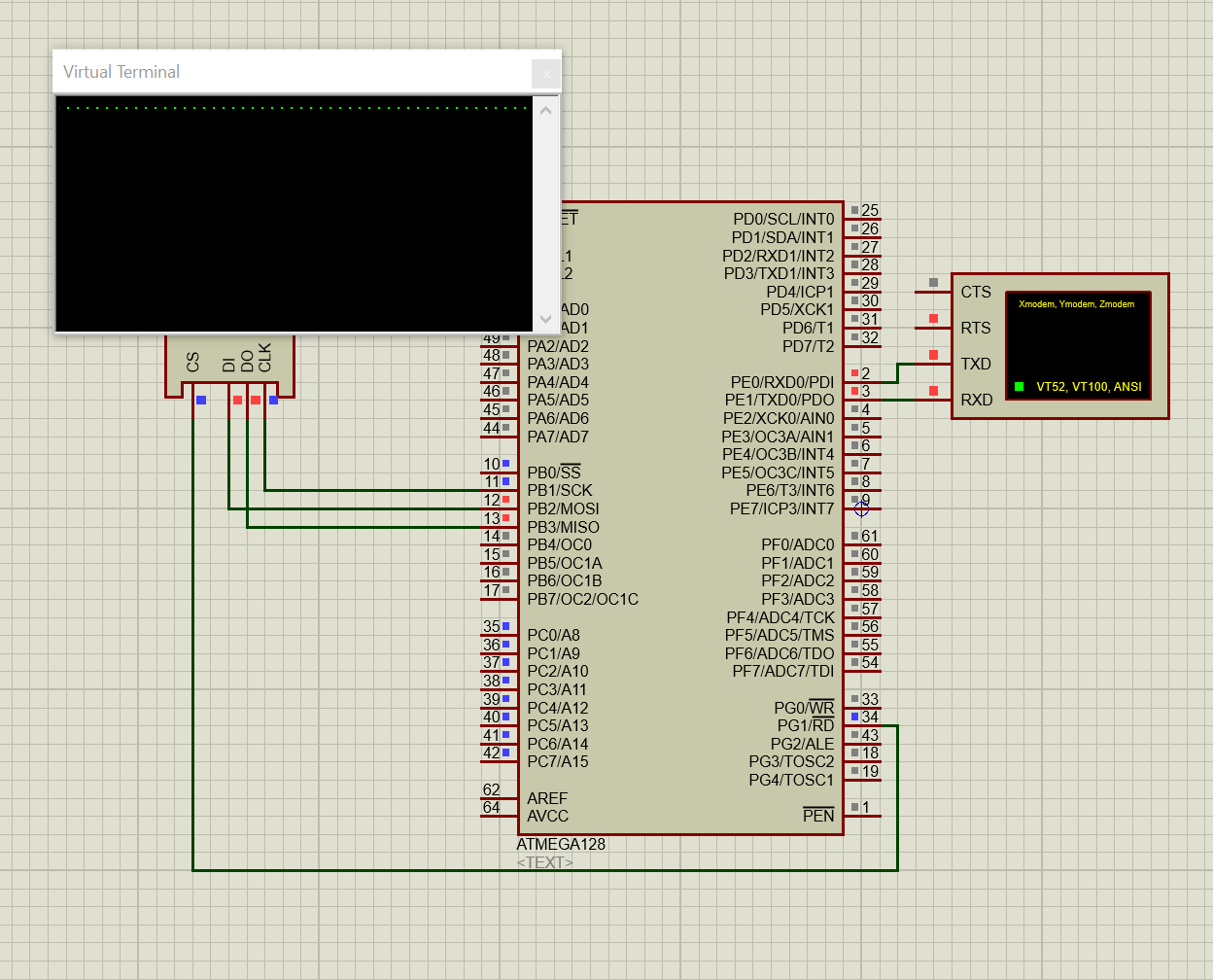
1. Using Proteus to create a circuit below:
2. Upload the sd.mmc to the MMC card



1. Study code provided

*Full Code is in the appendix part.*

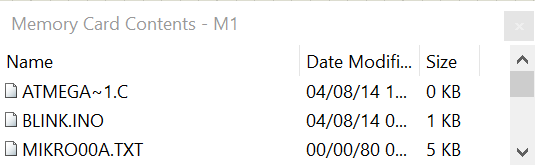
**Result:**



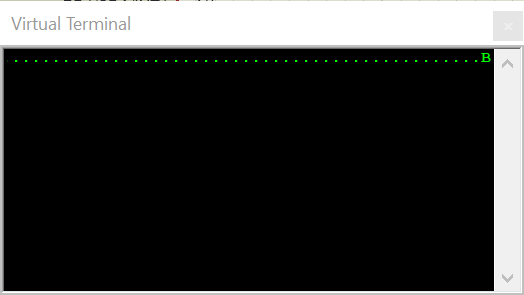
In this part, we’ll be working with data in SD card.

**Action of Code we’ve seen:**

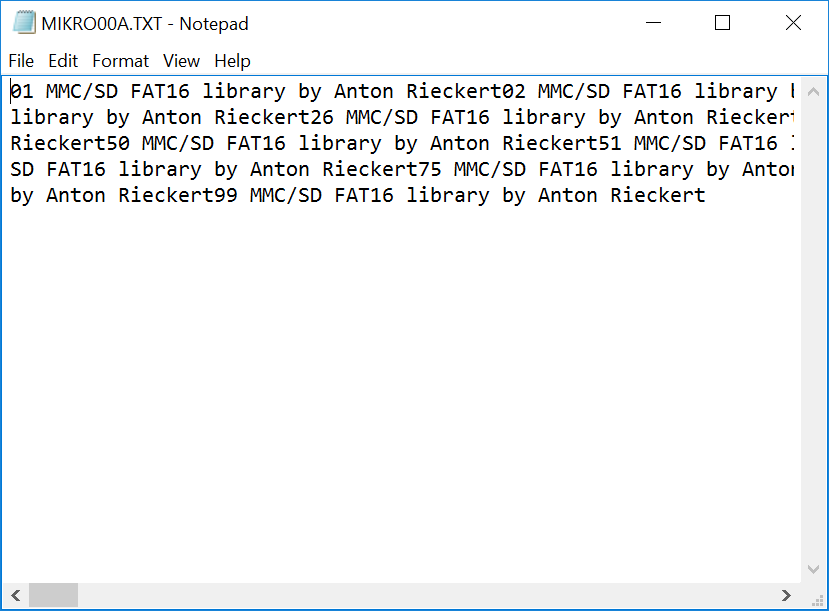
**M\_Create\_New\_File()**



Starting by create a file “MIKRO00A.TXT”

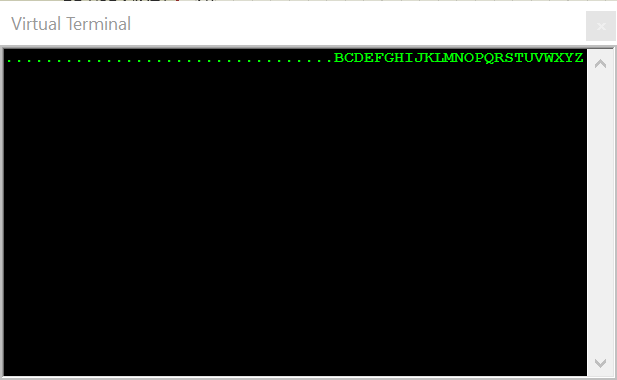


After that it will show result ‘.’ in virtual terminal.

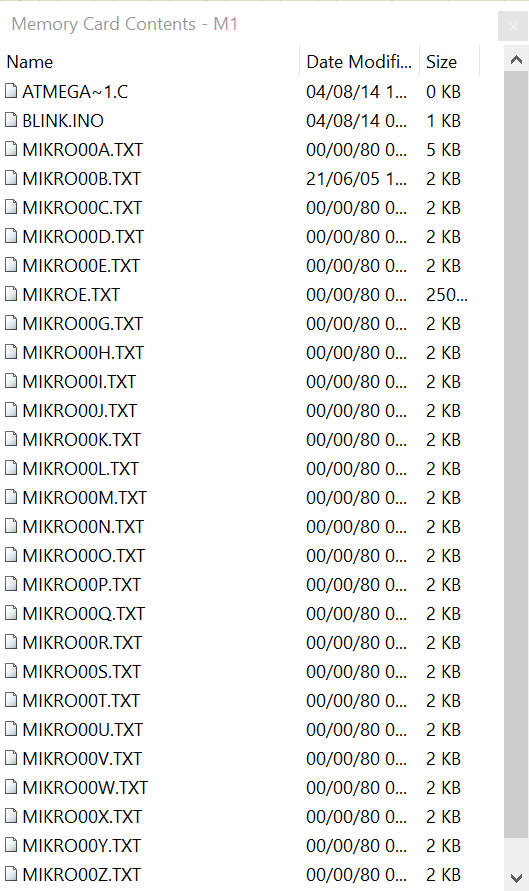


The content in “MIKRO00A.TXT” file.

**M\_Create\_Multiple\_Files()**

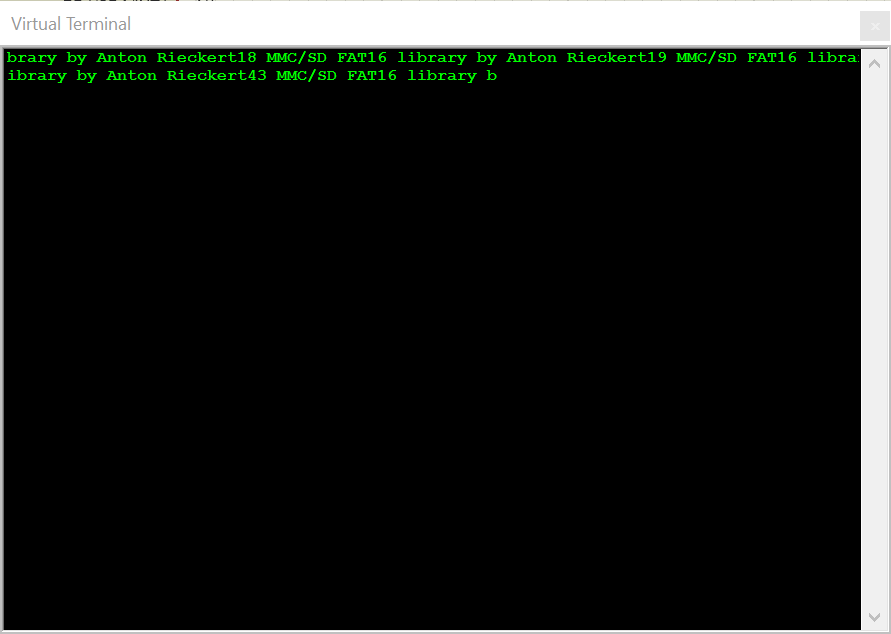


It will print letter ‘B’ to letter ‘Z’.



And also create files with index ‘B’ to ‘Z’ (MIKRO00x.txt , x is index).

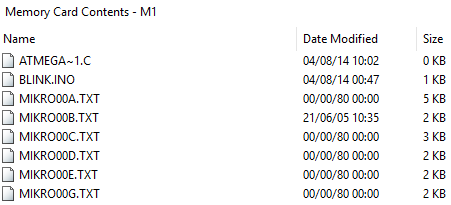
**M\_Open\_File\_Read()**



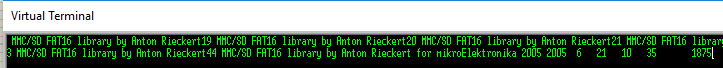
Read content from file.

**M\_Delete\_File()**

MIKRO00F.TXT will be deleted from SD card

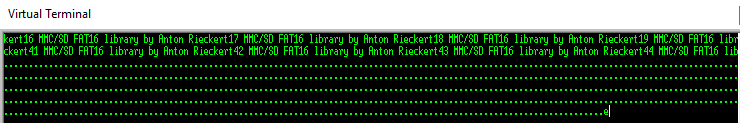


**When M\_Test\_File\_Exist function works**



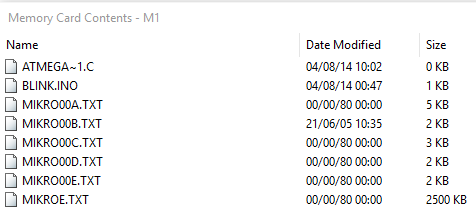
It displays date and time that created file and file’s size

**M\_Create\_Swap\_File works and end of the program**



It displays ‘.’ (dots) for each loop it goes and displays e when everything is finished.

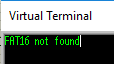
Also the file “MIKROE.TXT” is created.



And its content is not quite clear about what is it as shown.

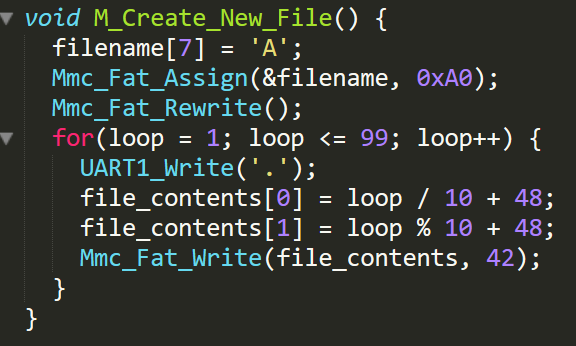


Moreover if we can’t find SD card that is FAT16 it will displays

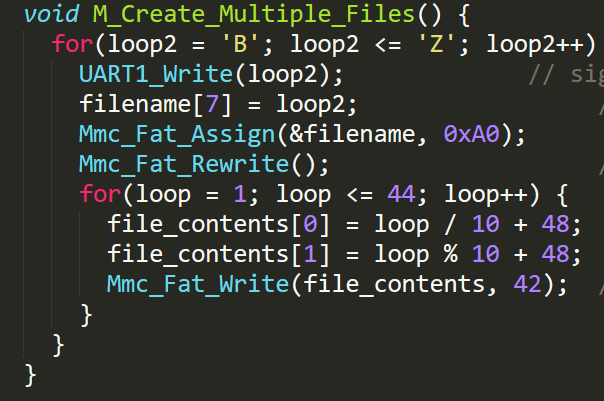


**Discussion:**

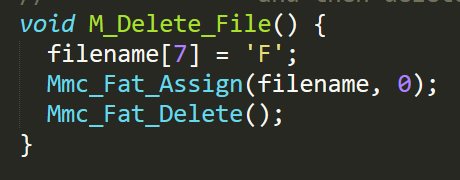
**Part of code for working with SD card:**



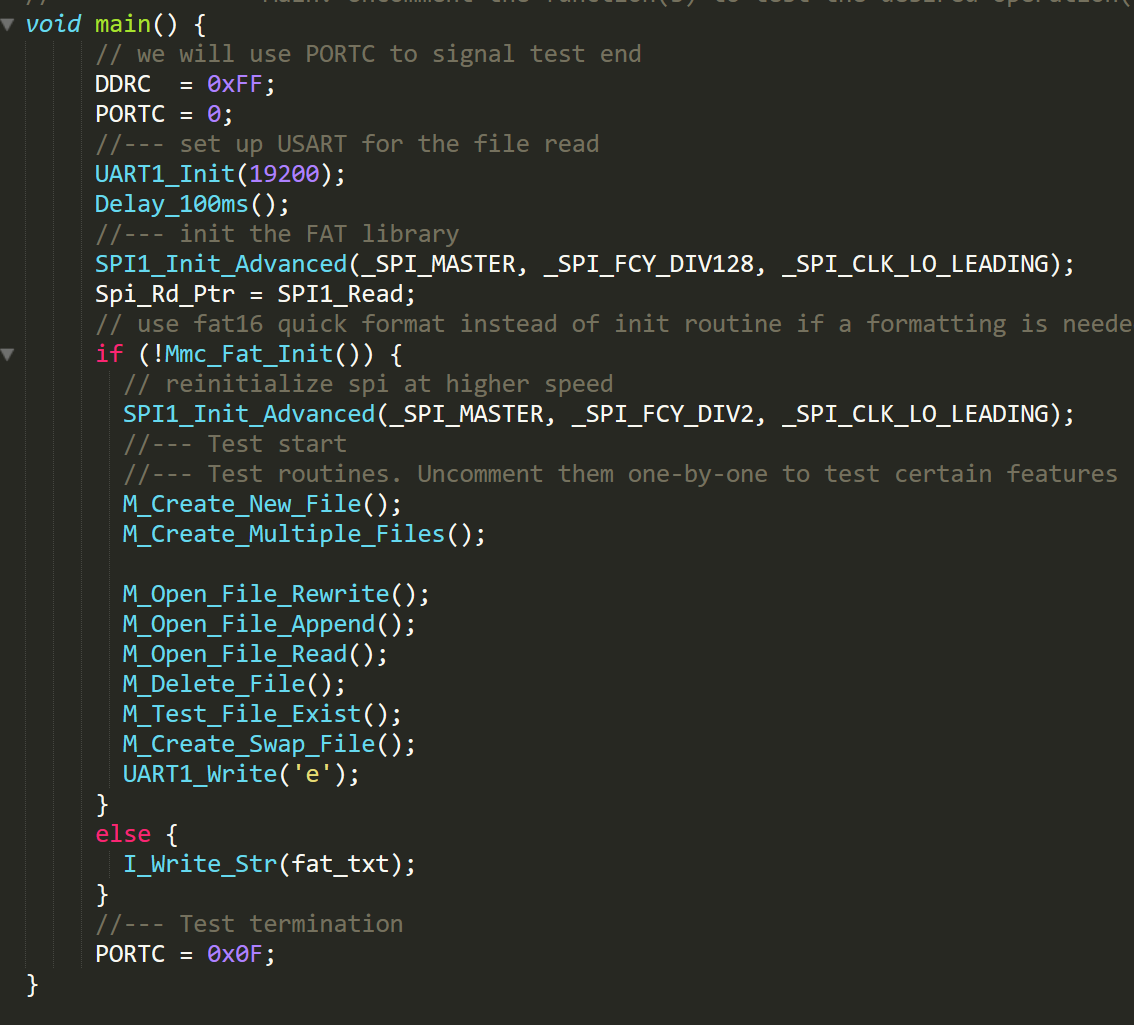
Create a new file with letter ‘A’ in filename 7th character (substitute x) and check condition, if there have file exits, create new one. After that write “.” in virtual terminal. In last section, in loop for, we need to change the content in file to me another character by using ASCII code.



In this function, we create a multiple files start with letter ‘B’ to letter ‘Z’ by using loop then change content in file same as previous function.



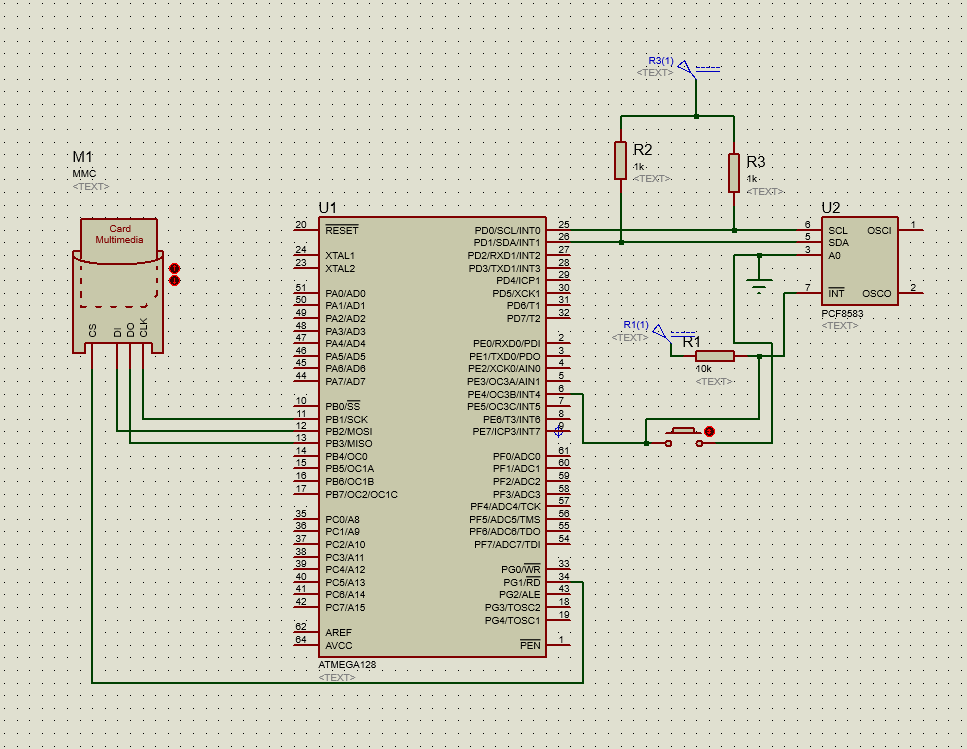
This is a delete function. We delete file by substituting file with file name with letter ‘F’ after that delete file.



In main function, we need to initial each value including baud rate. Overall, we call function which we have created already. When we called all function, it’ll print ‘e’ by UART in virtual terminal.

**Challenge**: Create a Time Logger with one switch. The Time Logger is a device that a person can come to push a button, then it will record at time on a SD card.

Here is the circuit for Time Logger:



And Full Code in the appendix.

Main part of Code

1. void interrupt\_ISR(void) org IVT\_ADDR\_INT4 {
2. i2c\_start();
3. i2c\_write(0xA0);
4. i2c\_write(0x02);
5. i2c\_start();
6. i2c\_write(0xA1);
7. second = i2c\_readwithack();
8. minute = i2c\_readwithack();
9. hour = (0x3f&i2c\_readwithoutack());
10. i2c\_stop();
11. Mmc\_Fat\_Assign("TIMELOG.txt", 0);
13. file\_contents[0] = hour / 10 + '0';
14. file\_contents[1] = hour % 10 + '0';
15. file\_contents[3] = minute / 16 + '0';
16. file\_contents[4] = minute % 16 + '0';
17. file\_contents[6] = second / 16 + '0';
18. file\_contents[7] = second % 16 + '0';
19. file\_contents[8] = 13;
20. file\_contents[9] = 10; // 13 and 10 last 2 word is for newline code
22. Mmc\_Fat\_Append();
23. Mmc\_Fat\_Write(file\_contents, 10);
24. }
26. void main() {
28. DDRD  = 0xFF;
30. Delay\_100ms();
32. SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV128, \_SPI\_CLK\_LO\_LEADING);
33. Spi\_Rd\_Ptr = SPI1\_Read;
35. EIMSK = (1<< INT4);
36. SREG\_I\_bit = 1;
38. i2c\_init();
40. if (!Mmc\_Fat\_Init()) {
42. SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV2, \_SPI\_CLK\_LO\_LEADING);
43. M\_Create\_New\_File();
45. while(1);
46. }
47. }

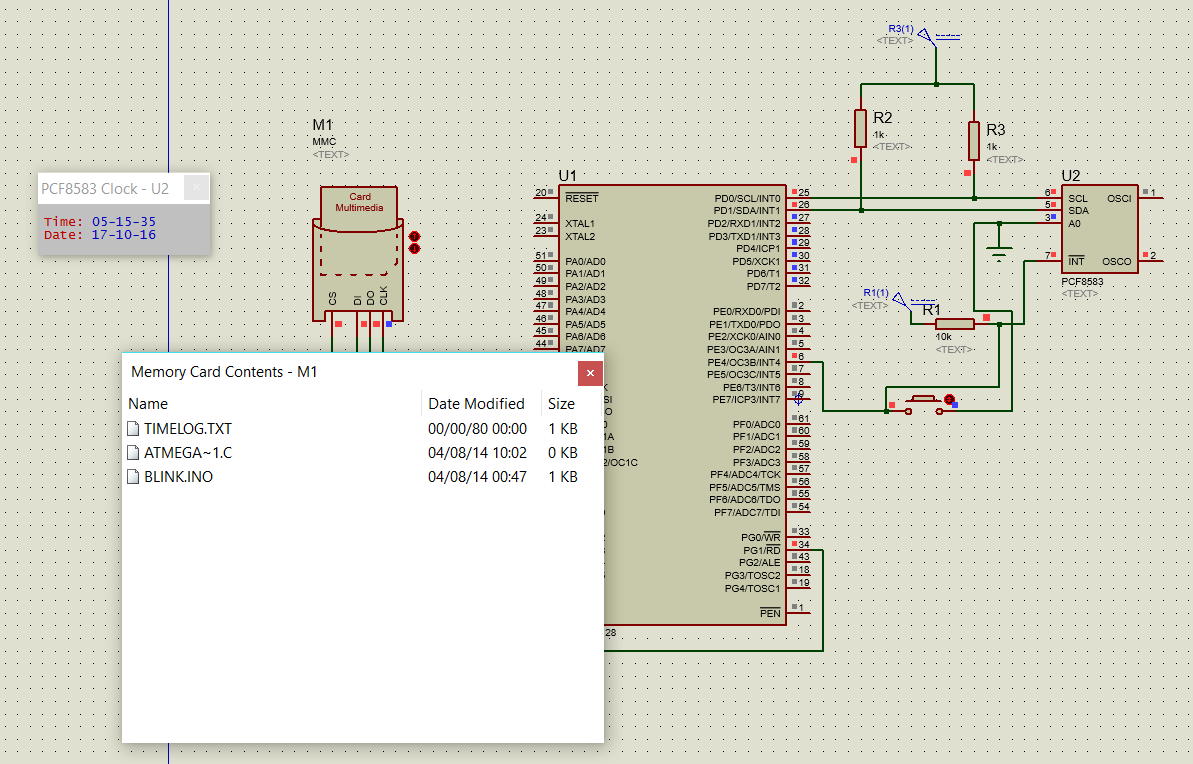
For this challenge we are using some part of code and knowledge from part1 and part2.

Steps to make Time Logger

1. Initial SPI and I2C .
2. Open file and waiting for an external interrupt.
3. When the interrupt comes, read data from RTC and Write data to SD Card
4. Waiting for external interrupt again, and loop back to 3

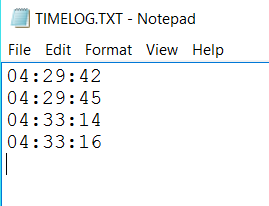
So we just integrate parts of code and add external interrupt for triggering the Time Logger

**Result**



Time Logger create file “TIMELOG.txt” that keep tracking which time the button is pressed

The content of file is as shown below:



**Conclusion**

From first 2 part of this LAB7 we have done working with RTC module that use a I2C protocol to communicate to others IC using SDA and SCL port.

And also working with MMC SD card using SPI to communicate with the MMC module to working with file in SD card

And the last part we integrate both part1 and part2 to create Time Logger that keep tracking what time the button is pressed

**Appendix**

|  |
| --- |
| **LAB7 part2 – SPI WITH SD CARD** |

|  |  |
| --- | --- |
|  | #include <built\_in.h> |
|  |  |
|  | sbit Mmc\_Chip\_Select at PORTG.B1; |
|  | sbit Mmc\_Chip\_Select\_Direction at DDRG.B1; |
|  |  |
|  | *char* |
|  | fat\_txt[20] = "FAT16 not found", |
|  | file\_contents[50] = "XX MMC/SD FAT16 library by Anton Rieckert\n"; |
|  |  |
|  | *char* |
|  | filename[14] = "MIKRO00xTXT";          // File names |
|  | *unsigned* *short* |
|  | tmp, caracter, loop, loop2; |
|  | *unsigned* *long* |
|  | i, size; |
|  |  |
|  | *char* Buffer[512]; |
|  |  |
|  | //I-I-I--------- Writes string to USART |
|  | *void* I\_Write\_Str(*char* \**ostr*) { |
|  | *unsigned* *short* i; |
|  |  |
|  | i = 0; |
|  | while (ostr[i]) { |
|  | UART1\_Write(ostr[i++]); |
|  | } |
|  | UART1\_Write(0x0A); |
|  | } |
|  |  |
|  | //M-M-M--------- Creates new file and writes some data to it |
|  | *void* M\_Create\_New\_File() { |
|  | filename[7] = 'A'; |
|  | Mmc\_Fat\_Assign(&filename, 0xA0);      // Will not find file and then create file |
|  | Mmc\_Fat\_Rewrite();                    // To clear file and start with new data |
|  | for(loop = 1; loop <= 99; loop++) {   //  We want 5 files on the MMC card |
|  | UART1\_Write('.'); |
|  | file\_contents[0] = loop / 10 + 48; |
|  | file\_contents[1] = loop % 10 + 48; |
|  | Mmc\_Fat\_Write(file\_contents, 42);   // write data to the assigned file |
|  | } |
|  | } |
|  |  |
|  | //M-M-M--------- Creates many new files and writes data to them |
|  | *void* M\_Create\_Multiple\_Files() { |
|  | for(loop2 = 'B'; loop2 <= 'Z'; loop2++) { |
|  | UART1\_Write(loop2);             // signal the progress |
|  | filename[7] = loop2;                 // set filename |
|  | Mmc\_Fat\_Assign(&filename, 0xA0);        // find existing file or create a new one |
|  | Mmc\_Fat\_Rewrite();                   // To clear file and start with new data |
|  | for(loop = 1; loop <= 44; loop++) { |
|  | file\_contents[0] = loop / 10 + 48; |
|  | file\_contents[1] = loop % 10 + 48; |
|  | Mmc\_Fat\_Write(file\_contents, 42);  // write data to the assigned file |
|  | } |
|  | } |
|  | } |
|  |  |
|  | //M-M-M--------- Opens an existing file and rewrites it |
|  | *void* M\_Open\_File\_Rewrite() { |
|  | filename[7] = 'C'; |
|  | Mmc\_Fat\_Assign(&filename, 0); |
|  | Mmc\_Fat\_Rewrite(); |
|  | for(loop = 1; loop <= 55; loop++) { |
|  | file\_contents[0] = loop / 10 + 64; |
|  | file\_contents[1] = loop % 10 + 64; |
|  | Mmc\_Fat\_Write(file\_contents, 42);    // write data to the assigned file |
|  | } |
|  | } |
|  |  |
|  | //M-M-M--------- Opens an existing file and appends data to it |
|  | //               (and alters the date/time stamp) |
|  | *void* M\_Open\_File\_Append() { |
|  | filename[7] = 'B'; |
|  | Mmc\_Fat\_Assign(&filename, 0); |
|  | Mmc\_Fat\_Set\_File\_Date(2005,6,21,10,35,0); |
|  | Mmc\_Fat\_Append();                                    // Prepare file for append |
|  | Mmc\_Fat\_Write(" for mikroElektronika 2005\n", 27);   // Write data to assigned file |
|  | }//~ |
|  |  |
|  | //M-M-M--------- Opens an existing file, reads data from it and puts it to USART |
|  | *void* M\_Open\_File\_Read() { |
|  | filename[7] = 'B'; |
|  | Mmc\_Fat\_Assign(&filename, 0); |
|  | Mmc\_Fat\_Reset(&size);                 // To read file, procedure returns size of file |
|  | for (i = 1; i <= size; i++) { |
|  | Mmc\_Fat\_Read(&caracter); |
|  | UART1\_Write(caracter);         // Write data to USART |
|  | } |
|  | } |
|  |  |
|  | //M-M-M--------- Deletes a file. If file doesn't exist, it will first be created |
|  | //               and then deleted. |
|  | *void* M\_Delete\_File() { |
|  | filename[7] = 'F'; |
|  | Mmc\_Fat\_Assign(filename, 0); |
|  | Mmc\_Fat\_Delete(); |
|  | } |
|  |  |
|  | //M-M-M--------- Tests whether file exists, and if so sends its creation date |
|  | //               and file size via USART |
|  | *void* M\_Test\_File\_Exist() { |
|  | *unsigned* *long* fsize; |
|  | *unsigned* *int* year; |
|  | *unsigned* *short* month, day, hour, minute; |
|  | *unsigned* *char* outstr[12]; |
|  |  |
|  | filename[7] = 'B';       //uncomment this line to search for file that DOES exists |
|  | //  filename[7] = 'F';       //uncomment this line to search for file that DOES NOT exist |
|  | if (Mmc\_Fat\_Assign(filename, 0)) { |
|  | //--- file has been found - get its date |
|  | Mmc\_Fat\_Get\_File\_Date(&year, &month, &day, &hour, &minute); |
|  | WordToStr(year, outstr); |
|  | I\_Write\_Str(outstr); |
|  | ByteToStr(month, outstr); |
|  | I\_Write\_Str(outstr); |
|  | WordToStr(day, outstr); |
|  | I\_Write\_Str(outstr); |
|  | WordToStr(hour, outstr); |
|  | I\_Write\_Str(outstr); |
|  | WordToStr(minute, outstr); |
|  | I\_Write\_Str(outstr); |
|  | //--- get file size |
|  | fsize = Mmc\_Fat\_Get\_File\_Size(); |
|  | LongToStr((*signed* *long*)fsize, outstr); |
|  | I\_Write\_Str(outstr); |
|  | } |
|  | else { |
|  | //--- file was not found - signal it |
|  | UART1\_Write(0x55); |
|  | Delay\_ms(1000); |
|  | UART1\_Write(0x55); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | //-------------- Tries to create a swap file, whose size will be at least 100 |
|  | //               sectors (see Help for details) |
|  | *void* M\_Create\_Swap\_File() { |
|  | *unsigned* *int* i; |
|  |  |
|  | for(i=0; i<512; i++) |
|  | Buffer[i] = i; |
|  |  |
|  | size = Mmc\_Fat\_Get\_Swap\_File(5000, "mikroE.txt", 0x20);   // see help on this function for details |
|  |  |
|  | if (size) { |
|  | LongToStr((*signed* *long*)size, fat\_txt); |
|  | I\_Write\_Str(fat\_txt); |
|  |  |
|  | for(i=0; i<5000; i++) { |
|  | Mmc\_Write\_Sector(size++, Buffer); |
|  | UART1\_Write('.'); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | //-------------- Main. Uncomment the function(s) to test the desired operation(s) |
|  | *void* main() { |
|  | // we will use PORTC to signal test end |
|  | DDRC  = 0xFF; |
|  | PORTC = 0; |
|  | //--- set up USART for the file read |
|  | UART1\_Init(19200); |
|  | Delay\_100ms(); |
|  | //--- init the FAT library |
|  | SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV128, \_SPI\_CLK\_LO\_LEADING); |
|  | Spi\_Rd\_Ptr = SPI1\_Read; |
|  | // use fat16 quick format instead of init routine if a formatting is needed |
|  | if (!Mmc\_Fat\_Init()) { |
|  | // reinitialize spi at higher speed |
|  | SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV2, \_SPI\_CLK\_LO\_LEADING); |
|  | //--- Test start |
|  | //--- Test routines. Uncomment them one-by-one to test certain features |
|  | M\_Create\_New\_File(); |
|  | M\_Create\_Multiple\_Files(); |
|  |  |
|  | M\_Open\_File\_Rewrite(); |
|  | M\_Open\_File\_Append(); |
|  | M\_Open\_File\_Read(); |
|  | M\_Delete\_File(); |
|  | M\_Test\_File\_Exist(); |
|  | M\_Create\_Swap\_File(); |
|  | UART1\_Write('e'); |
|  | } |
|  | else { |
|  | I\_Write\_Str(fat\_txt); |
|  | } |
|  | //--- Test termination |
|  | PORTC = 0x0F; |
|  | }   |  | | --- | | **Challenge: Time Logger** | |

1. #include <built\_in.h>
3. sbit Mmc\_Chip\_Select at PORTG.B1;
4. sbit Mmc\_Chip\_Select\_Direction at DDRG.B1;
6. char hour, minute,second;
8. char file\_contents[10] = "XX:XX:XX**\n**";
10. void M\_Create\_New\_File() {
12. Mmc\_Fat\_Assign("TIMELOG.txt", 0xA0);
13. Mmc\_Fat\_Rewrite();
14. }
16. void i2c\_write(unsigned char data2) {
18. TWDR = data2;
19. TWCR = 1<<(TWINT)|1<<(TWEN);
20. while(!(TWCR&1<<(TWINT)));
21. }
23. unsigned int i2c\_readwithoutack(void) {
25. TWCR = 1<<(TWINT) | 1<<(TWEN);
26. while(!(TWCR&1<<(TWINT)));
27. return TWDR;
28. }
29. unsigned int i2c\_readwithack(void) {
31. TWCR = 1<<(TWINT) | 1<<(TWEN)|1<<(TWEA);
32. while(!(TWCR&1<<(TWINT)));
33. return TWDR;
34. }
36. void i2c\_init(void) {
38. TWSR = 0x00;
39. TWBR = 0x47;
40. TWCR = 0x04;
41. }
43. void i2c\_start(void) {
45. TWCR = 1<<(TWINT) | 1<<(TWSTA)|1<<(TWEN);
46. while((TWCR &1<<(TWINT))==0);
47. }
49. void i2c\_stop(void) {
51. TWCR = 1<<(TWINT)|1<<(TWEN)|1<<(TWSTO);
52. }
54. void interrupt\_ISR(void) org IVT\_ADDR\_INT4
55. {
56. i2c\_start();
57. i2c\_write(0xA0);
58. i2c\_write(0x02);
59. i2c\_start();
60. i2c\_write(0xA1);
61. second = i2c\_readwithack();
62. minute = i2c\_readwithack();
63. hour = (0x3f&i2c\_readwithoutack());
64. i2c\_stop();
65. Mmc\_Fat\_Assign("TIMELOG.txt", 0);
67. file\_contents[0] = hour / 10 + '0';
68. file\_contents[1] = hour % 10 + '0';
69. file\_contents[3] = minute / 16 + '0';
70. file\_contents[4] = minute % 16 + '0';
71. file\_contents[6] = second / 16 + '0';
72. file\_contents[7] = second % 16 + '0';
73. file\_contents[8] = 13;
74. file\_contents[9] = 10;
76. Mmc\_Fat\_Append();
77. Mmc\_Fat\_Write(file\_contents, 10);
78. }
80. void main() {
82. DDRD  = 0xFF;
84. Delay\_100ms();
86. SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV128, \_SPI\_CLK\_LO\_LEADING);
87. Spi\_Rd\_Ptr = SPI1\_Read;
89. EIMSK = (1<< INT4);
90. SREG\_I\_bit = 1;
92. i2c\_init();
94. if (!Mmc\_Fat\_Init()) {
96. SPI1\_Init\_Advanced(\_SPI\_MASTER, \_SPI\_FCY\_DIV2, \_SPI\_CLK\_LO\_LEADING);
97. M\_Create\_New\_File();
99. while(1);
100. }
101. }