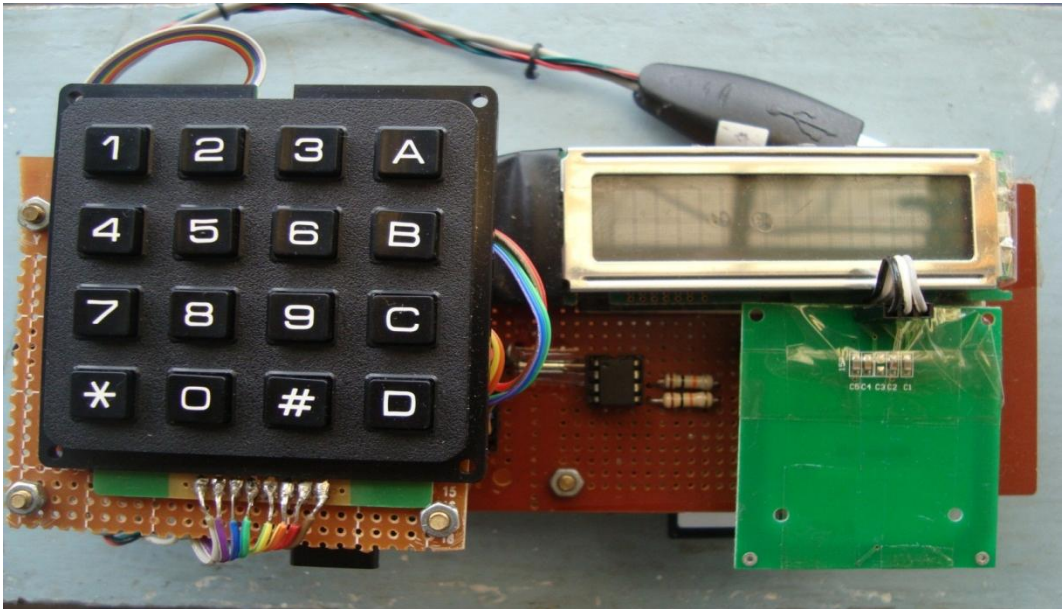
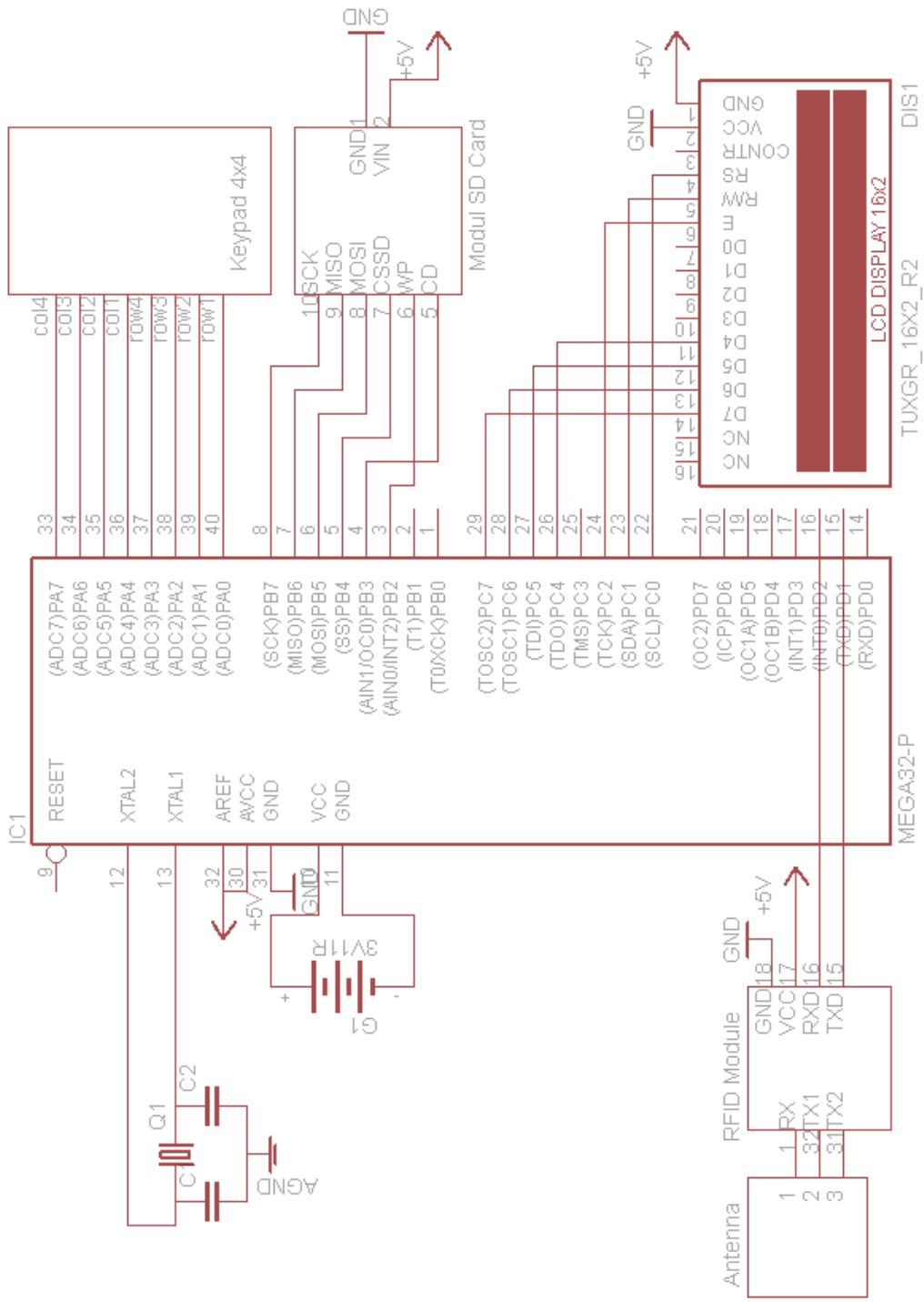


**LAMPIRAN A**  
**FOTO ALAT**



**LAMPIRAN B**

**SKEMATIK PERANGKAT PEMBACA DAN  
PENYIMPAN DATA RFID PORTABLE**



**LAMPIRAN C**  
**DATASHEET**

# EMS (Embedded Module Series)

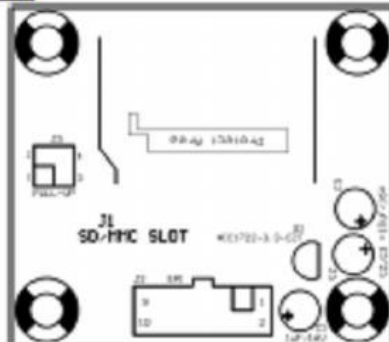
## SD/MMC/FRAM

**EMS SD/MMC/FRAM** merupakan suatu modul untuk mempermudah antarmuka antara SD Card (atau MMC) dan mikrokontroler dengan tegangan kerja +5 VDC. SD Card (atau MMC) dapat digunakan sebagai memori yang dapat diganti dengan mudah sehingga memudahkan dalam ekspansi ke kapasitas memori yang lebih besar. Tersedia Ferroelectric Nonvolatile RAM (FRAM) yang dapat digunakan sebagai buffer sementara dalam mengakses SD Card (atau MMC) atau sebagai tempat penyimpanan data lain. Modul ini dapat digunakan antara lain sebagai penyimpan data pada sistem absensi, sistem antrian, atau aplikasi datalogging lainnya.

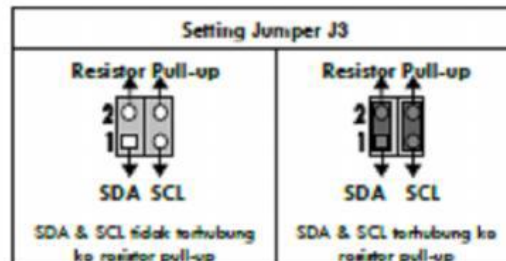
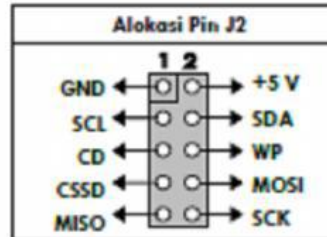
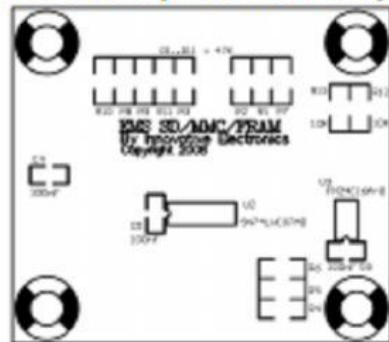
### Spesifikasi Hardware

1. Tegangan supply +5 VDC.
2. Jenis kartu yang didukung: SD Card (dan MMC).
3. Antarmuka SD Card (dan MMC) dengan mikrokontroler secara SPI.
4. Tersedia 2 KByte Ferroelectric Nonvolatile RAM FM24C16.
5. Antarmuka FRAM dengan mikrokontroler secara Two-Wire Interface.
6. Tersedia contoh aplikasi untuk DT-S1™ Low Cost Series dan DT-AVR Low Cost Series dalam bahasa BASIC untuk MCS-51® (BASCOM-8051®) dan bahasa C untuk AVR® (CodeVisionAVR®).
7. Kompatibel dengan DT-S1™ Low Cost Series dan DT-AVR Low Cost Series. Mendukung DT-S1™ Minimum System (MinSys) ver 3.0, DT-S1™ PutraFuz, dan lain-lain.

### Tata Letak



Terdapat kesalahan cetak pada PCB: J2 ISP seharusnya J2 SPI.



Jumper J3 digunakan untuk resistor pull-up SDA dan SCL. Apabila modul terhubung ke jaringan Two-Wire Interface, maka dalam satu jaringan tersebut hanya perlu memasang pull-up pada salah satu modul saja.

Pin	Nama	Fungsi pada Modul	Keterangan
1	GND	Input	Referensi Ground
2	+5 V	Input	Terhubung ke Sumber Tegangan +5 VDC
3	SCL	Input	Serial Clock untuk akses FRAM
4	SDA	Input/Output	Serial Data untuk transaksi data dari/ke FRAM
5	CD	Output	Card Detect, berlogika 0 jika ada kartu yang dimasukkan, berlogika 1 jika tidak ada kartu
6	WP	Output	Write Protect, berlogika 0 jika saklar pada SD Card tidak berada pada posisi dikunci, berlogika 1 jika SD Card dalam posisi dikunci
7	CSSD	Input	Chip Select, diberi logika 0 untuk mengakses SD Card, diberi logika 1 jika tidak mengakses SD Card
8	MOSI	Input	Jalur data masuk ke SD Card
9	MISO	Output	Jalur data keluar dari SD Card
10	SCK	Input	Jalur clock dari mikrokontroler untuk mengakses SD Card

### Isi CD

1. Contoh Aplikasi dengan CodeVisionAVR® dan BASCOM-8051®.
2. Datasheet.
3. Manual EMS SD/MMC/FRAM.
4. Website Inovative Electronics.

### Prosedur Testing

Ikuti petunjuk pada contoh aplikasi AN SD BAS51.PDF untuk MCS-51® (BASCOM-8051®) dan AN SD CVAVR.PDF untuk AVR® (CodeVisionAVR®). Tersedia juga AN SD FAT.PDF untuk pengenalan awal terhadap protokol SD Card dan format FAT16.



# EMS SD/MMC/FRAM *Application Note*

## Akses SD Card & FRAM Menggunakan AVR

Oleh: Tim IE

**S**ecure Digital (SD) atau MultiMedia Card (MMC) seringkali digunakan sebagai sarana penyimpan data pada Personal Digital Assistant (PDA), kamera digital, dan telepon seluler (ponsel). AN Ini akan menggunakan EMS SD/MMC/FRAM sebagai sarana penyimpanan data dengan menggunakan SD card dan FRAM yang tersedia.

Aplikasi ini terdiri dari 4 program:

1. DT-AVR Low Cost Micro System membaca parameter format data SD Card lalu mengirimkannya secara UART ke komputer.
  - a. Menggunakan EEPROM ATmega8535 sebagai buffer data.
  - b. Menggunakan FRAM sebagai buffer data.
2. Berdasar parameter yang sudah dibaca dan sudah dimasukkan ke program, DT-AVR Low Cost Micro System akan menulis sebuah file beserta isinya ke dalam SD Card.
  - a. Menggunakan EEPROM ATmega8535 sebagai buffer data.
  - b. Menggunakan FRAM sebagai buffer data.

Bahasa pemrograman yang digunakan adalah bahasa C (CodeVisionAVR<sup>®</sup>).

Modul yang diperlukan:

- 1 DT-AVR Low Cost Micro System,
- 1 EMS SD/MMC/FRAM,
- 1 SD Card dengan format FAT16 (umumnya berkapasitas antara 32 MB hingga 2 GB)

Pastikan SD card yang digunakan tidak menyimpan data penting atau terproteksi karena program pada AN ini akan menghapus/merusak isi sebelumnya.

**A**dapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



Gambar 1  
Blok Diagram

**H**ubungan antara modul-modul tersebut terdapat pada tabel berikut:

DT-AVR Low Cost Micro System	EMS SD/MMC/FRAM
GND (PORTC pin 1)	GND (J2 pin 1)
VCC (PORTC pin 2)	VCC (J2 pin 2)
PC.0 (PORTC pin 3)**	SCL (J2 pin 3)
PC.1 (PORTC pin 4)**	SDA (J2 pin 4)
PB.2 (PORTB pin 5)*	CD (J2 pin 5)
PB.3 (PORTB pin 6)*	WP (J2 pin 6)
PB.4 (PORTB pin 7)*	CSSD (J2 pin 7)
PB.5 (PORTB pin 8)**	MOSI (J2 pin 8)
PB.6 (PORTB pin 9)**	MISO (J2 pin 9)
PB.7 (PORTB pin 10)**	SCK (J2 pin 10)

\* Pin ini tidak mutlak dan dapat diganti pin lain tetapi harus mengubah program

\*\*Pin ini mutlak digunakan jika menggunakan komunikasi TWI dan SPI hardware

Tabel 1  
Hubungan DT-AVR Low Cost Micro System dengan EMS SD/MMC/FRAM

Jumper J4 & J5 (DT-AVR Low Cost Micro System) perlu diatur agar PD.0 dan PD.1 terhubung ke rangkaian UART RS-232 (posisi 1-2). Hubungkan DT-AVR Low Cost Micro System ke komputer melalui kabel serial yang tersedia.

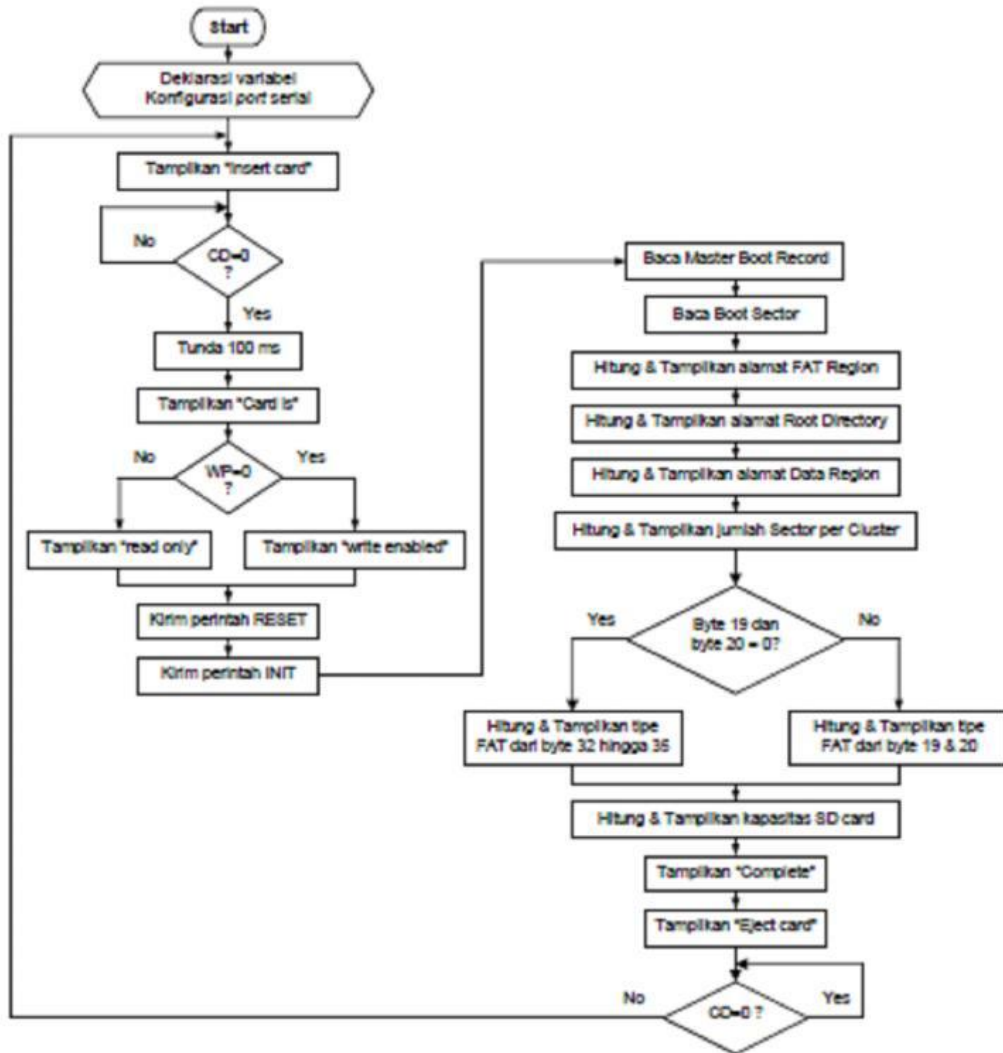


Pada komputer, gunakan program Terminal pada CodeVisionAVR<sup>®</sup>, HyperTerminal<sup>®</sup>, Terminal<sup>®</sup>, atau program terminal lainnya dengan baud rate 9600, 8 bit data, 1 bit stop, tanpa bit parity, dan tanpa flow control.

Setelah semua rangkaian dan catu daya terhubung dengan benar, programlah Read w EEPROM – cvavr.hex atau Read w RAM – cvavr.hex DT-AVR Low Cost Micro System menggunakan DT-HIQ AVR In System Programmer atau divals programmer lain yang mendukung.

Jika pemrograman ISP terganggu, coba lepas SD Card atau lepas modul EMS SD/MMC/FRAM atau ubah program dan pindahkan koneksi ke port lain.

**F**lowchart dari program Read w EEPROM – cvavr.prj dan Read w RAM – cvavr.prj adalah sebagai berikut:



Gambar 2  
Flowchart Program Read w EEPROM – cvavr.prj dan Read w RAM – cvavr.prj

**C**ara kerja program secara garis besar adalah sebagai berikut:

1. Pertama program melakukan deklarasi variabel yang akan digunakan untuk menampung data dan parameter yang berhubungan dengan SD card. Program juga melakukan konfigurasi port serial pada baud rate 9600, 8 bit data, 1 bit stop, tanpa bit parity, dan tanpa flow control.
2. Program mengirimkan "insert card" ke komputer lalu menunggu adanya kartu yang dimasukkan ke slot.
3. Setelah ada kartu yang dimasukkan ke slot (CD=0), program akan menunda 100 milidetik sebelum mengakses kartu. Hal ini bertujuan untuk memastikan bahwa kartu sudah tepat berada di tempatnya saat akan diakses. Nilai waktu tunda ini dapat diubah atau dihilangkan jika dirasa tidak diperlukan.
4. Program juga akan memeriksa apakah tuas pada SD card dalam posisi terkunci (WP=1) atau tidak (WP=0), dan menampilkan kondisinya pada layar terminal komputer.
5. Langkah awal sebelum mengakses SD card untuk pertama kalinya adalah mengirimkan perintah Reset dan Init ke SD card.
6. Karena SD card yang digunakan memiliki format FAT16, maka parameter yang harus dibaca disesuaikan dengan format FAT16. Yang pertama harus dilakukan adalah membaca Master Boot Record (berada di sektor 0) untuk mengetahui lokasi Boot Sector.
7. Lalu Boot Sector dibaca secara keseluruhan. Nilai pada alamat-alamat tertentu diambil dan dihitung sehingga didapat parameter antara lain: alamat FAT Region, alamat Root Directory, alamat Data Region, jumlah sector per cluster, tipe FAT, dan kapasitas SD card.
8. Program menampilkan pesan "Complete" lalu "Eject Card" dan menunggu kartu dikeluarkan.
9. Berikut ini tampilan yang muncul pada program terminal:

```
Insert card
Card is read only      atau   Card is write enabled
Reset - OK
Init - OK
FATregion: 59
RootDir: 81
DataRegion: 113
SectorCluster: 8
Type: FAT16
Size (MB): 30
Complete
Eject card
```

10. Setelah kartu dikeluarkan (CD=1), program kembali ke langkah nomor 2.

Pada program **Read w EEPROM – cvavr.prj**, data yang dibaca disimpan dalam variabel yang diletakkan pada EEPROM. Hal ini dikarenakan internal RAM Atmega8535 tidak mencukupi. Besarnya EEPROM sesuai dengan *buffer* yang diperlukan untuk akses SD card yaitu 512 byte sehingga proses menjadi mudah. Namun hal ini akan mengakibatkan usia pemakaian EEPROM menjadi cepat habis. Oleh karena itu, penggunaan EEPROM sebagai *buffer* tidak dianjurkan dan program ini hanya digunakan sebagai contoh saja.

Sedangkan pada program **Read w RAM – cvavr.prj**, data yang dibaca akan disimpan ke dalam variabel sementara dan akan langsung disimpan ke dalam FRAM. Internal RAM ATmega8535 yang digunakan hanya 256 byte yang disesuaikan dengan kapasitas 1 page FRAM. Oleh karena itu program membutuhkan 2 page FRAM saat mengakses SD card dimana 256 byte pertama diletakkan pada page 0 dan 256 byte kedua diletakkan pada page 1.

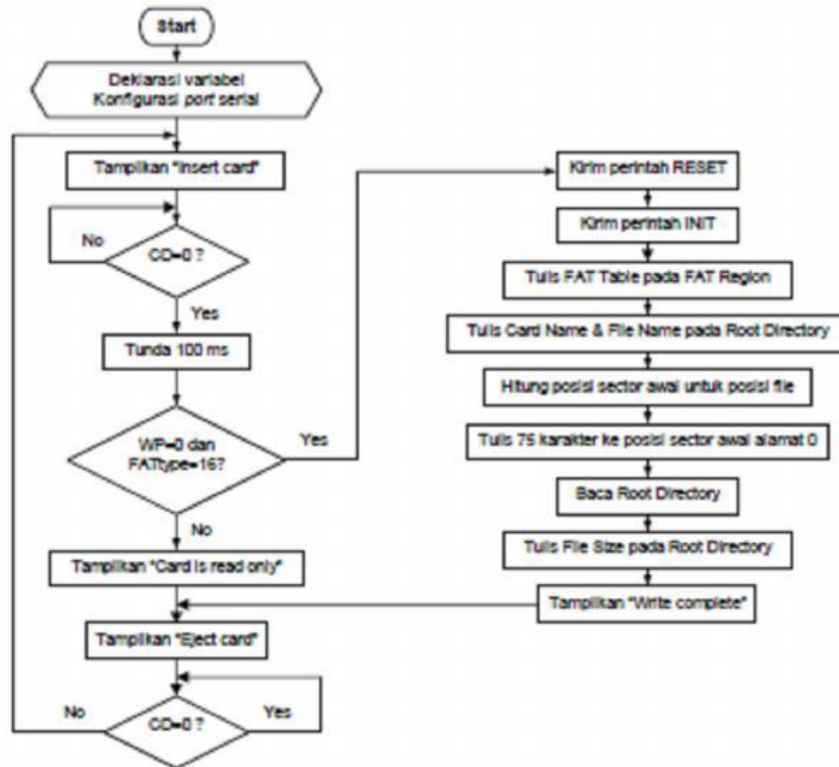
Setelah program **Read w EEPROM – cvavr.prj** dan **Read w RAM – cvavr.prj** berjalan normal dan menampilkan parameter seperti pada langkah 9, maka sesuaikan program **Write w EEPROM – cvavr.prj** dan **Write w RAM – cvavr.prj** dengan parameter tersebut. Contoh setelah baris program untuk deklarasi konstanta pada **Write w EEPROM – cvavr.c** dan **Write w RAM – cvavr.c** diganti:

```
unsigned long int FATregion=59;
unsigned long int RootDir=81;
unsigned long int DataRegion=113;
unsigned char SectorCluster=8;
unsigned char FATtype=16;
```

Periksa dan pastikan bahwa nilai yang dituliskan sama dengan nilai yang telah dibaca oleh program Read w EEPROM – cvavr.prj dan Read w RAM – cvavr.prj. Kesalahan atau perbedaan nilai dapat mengakibatkan format FAT16 pada SD card kacau dan tidak dapat diakses oleh card reader, kamera digital, PDA, dsb.

Setelah itu lakukan compile terhadap Write w EEPROM - cvavr.prj dan Write w RAM – cvavr.prj dan program ke DT-AVR Low Cost Micro System.

**F**lowchart dari program Write w EEPROM – cvavr.prj dan Write w RAM – cvavr.prj adalah sebagai berikut:



Gambar 3  
Flowchart Program Write w EEPROM – cvavr.prj dan Write w RAM – cvavr.prj

**C**ara kerja program secara garis besar adalah sebagai berikut:

1. Pertama program melakukan deklarasi variabel yang akan digunakan untuk menampung data dan parameter yang berhubungan dengan SD card. Program juga melakukan konfigurasi port serial pada baud rate 9600, 8 bit data, 1 bit stop, tanpa bit parity, dan tanpa flow control.
2. Program mengirimkan "insert card" ke komputer lalu menunggu adanya kartu yang dimasukkan ke slot (CD=0).
3. Setelah ada kartu yang dimasukkan ke slot (CD=0), program akan menunda 100 milidetik sebelum mengakses kartu. Hal ini bertujuan untuk memastikan bahwa kartu sudah tepat berada di tempatnya saat akan diakses. Nilai waktu tunda ini dapat diubah atau dihilangkan jika dirasa tidak diperlukan.
4. Jika kartu yang dimasukkan ke slot dalam posisi terkunci (WP=1), maka program akan menampilkan "Card is read only" dan "Eject card" lalu menunggu SD card dikeluarkan. Setelah SD card dikeluarkan (CD=1), program kembali ke langkah nomor 2.
5. Jika kartu yang dimasukkan tidak terkunci dan tipe FAT yang dimasukkan adalah FAT16, maka program akan melanjutkan akses ke SD card.

6. Langkah awal sebelum mengakses SD card untuk pertama kalinya adalah mengirimkan perintah Reset dan Init ke SD card.
7. Karena SD card yang digunakan memiliki format FAT16, maka proses menulis file harus disesuaikan dengan format FAT16. Program akan menulis tabel FAT pada FAT Region.
8. Program akan menulis Nama Kartu dan Nama File pada Root Directory.
9. Nilai Root Directory (yang terletak di FRAM page 2) akan dibaca untuk mengetahui posisi sector awal untuk file.
10. Isi file sebanyak 75 karakter dituliskan ke posisi sector awal mulai alamat 0.
11. Program membaca Root Directory.
12. Parameter ukuran file diubah lalu dituliskan kembali ke Root Directory.
13. Program menampilkan pesan "Write Complete" lalu "Eject Card" dan menunggu kartu dikeluarkan.
14. Berikut ini tampilan yang muncul pada program terminal jika SD card tidak terkunci dan proses penulisan berhasil:

```

Insert card
Init - OK
Writing:
FAT Table...
Card Name...
File Name...
File Contents...
File Size...
Write complete
Eject card

```

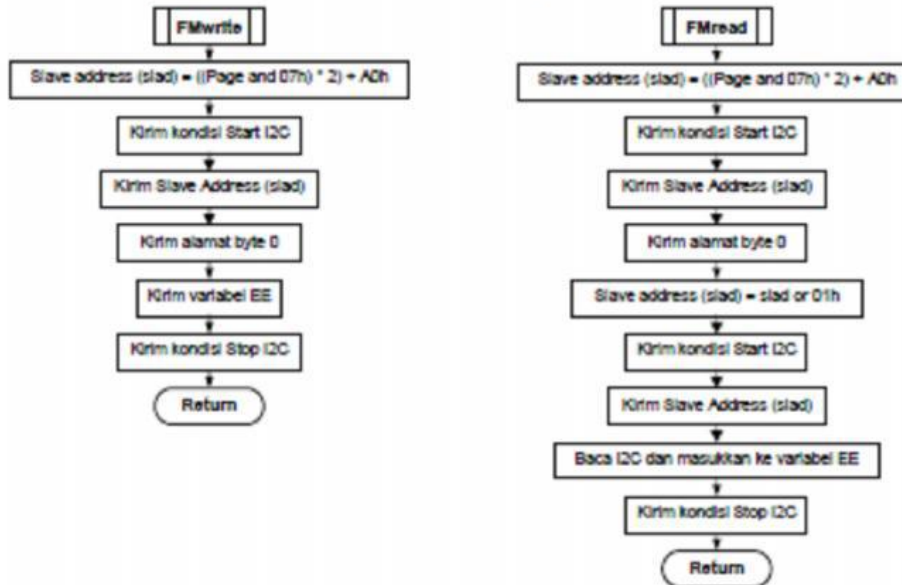
15. Setelah kartu dikeluarkan (CD=1), program kembali ke langkah nomor 2.

Setelah proses tulis selesai, maka SD card dapat dilepas lalu dibaca pada komputer menggunakan card reader atau pada PDA. Di dalamnya akan terdapat 1 file bernama TESTINGS.TXT sebesar 77 byte yang berisi: "0123456789; <->?@ABCDEFGHIJKLMNPQRSTUVWXYZ[ ]^\_`abodefghijklmnopqrstuvwxyz"

Pada program Write w EEPROM – cvavr.prj, data yang dibaca disimpan dalam variabel yang diletakkan pada EEPROM. Hal ini dikarenakan Internal RAM Atmega8535 tidak mencukupi. Besarnya EEPROM sesuai dengan buffer yang diperlukan untuk akses SD card yaitu 512 byte sehingga proses menjadi mudah. Namun hal ini akan mengakibatkan usia pemakaian EEPROM menjadi cepat habis. Oleh karena itu, penggunaan EEPROM sebagai buffer tidak dianjurkan dan program ini hanya digunakan sebagai contoh saja.

Sedangkan pada program Write w RAM – cvavr.prj, data yang dibaca akan disimpan ke dalam variabel sementara dan akan langsung disimpan ke dalam FRAM. Internal RAM ATmega8535 yang digunakan hanya 256 byte yang disesuaikan dengan kapasitas 1 page FRAM. Oleh karena itu program membutuhkan 2 page FRAM saat mengakses SD card. Pada program ini, page FRAM yang digunakan untuk proses baca dan tulis dibedakan sehingga jumlah page yang digunakan adalah 4 page. Pada proses baca, 256 byte pertama diletakkan pada page 0 dan 256 byte kedua diletakkan pada page 1. Pada proses tulis, 256 byte pertama diletakkan pada page 2 dan 256 byte kedua diletakkan pada page 3.

**F**lowchart dari rutin program FMread dan FMwrite adalah sebagai berikut:



Gambar 5  
Flowchart Rutin FMread dan FMwrite

**C**ara kerja rutin FMwrite adalah sebagai berikut:

1. Pertama rutin akan menghitung alamat slave sesuai dengan parameter page (pager) yang diberikan saat memanggil rutin.
2. Lalu rutin akan mengirim kondisi Start I2C yang diikuti dengan alamat slave yang telah dihitung.
3. Rutin selalu menulis mulai dari alamat 0 untuk tiap page sehingga data yang dikirim berikutnya adalah 0.
4. Semua isi variabel EE sebanyak 256 byte dituliskan ke FRAM.
5. Setelah semua data terkirim, rutin mengakhiri dengan mengirim kondisi Stop I2C.

**C**ara kerja rutin FMread adalah sebagai berikut:

1. Pertama rutin akan menghitung alamat slave sesuai dengan parameter page (pager) yang diberikan saat memanggil rutin.
2. Lalu rutin akan mengirim kondisi Start I2C yang diikuti dengan alamat slave yang telah dihitung.
3. Rutin selalu membaca mulai dari alamat 0 untuk tiap page sehingga data yang dikirim berikutnya adalah 0.
4. Lalu kondisi Start I2C akan dikirim yang diikuti dengan alamat slave untuk proses baca (bit 0 = 1).
5. Data sebanyak 256 byte pada page FRAM tersebut akan dibaca dan disimpan ke dalam variabel EE.
6. Setelah semua data dibaca, rutin mengakhiri dengan mengirim kondisi Stop I2C.

**L**isting program terdapat pada folder CVAVR.

**S**elamat berinovasi!

AVR is a registered trademark of Atmel.  
CodeVisionAVR is copyright by Pavel Halduc, HP InfoTech s.r.l.  
HyperTerminal is a copyright by Higlraeve Inc.  
Terminal is a copyright by Bray++.