DATA LOGGER PENGUKURAN SINYAL ANALOG MENGGUNAKAN SD CARD SEBAGAI MEDIA PENYIMPANAN

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:

DAHNIAL SYAUQY
NIM. 0510630027-63

DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK ELEKTRO
MALANG
2009

DATA LOGGER PENGUKURAN SINYAL ANALOG MENGGUNAKAN SD CARD SEBAGAI MEDIA PENYIMPANAN

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:

DAHNIAL SYAUQY
NIM. 0510630027-63

DOSEN PEMBIMBING

Pembimbing 1	Pembimbing 2
Ir. M. Julius St, MS.	**DR. Agung Darmawansyah, ST., MT**
NIP. 19540720 198203 1 002	**NIP. 19721218 199903 1 002**

LEMBAR PENGESAHAN

DATA LOGGER PENGUKURAN SINYAL ANALOG MENGGUNAKAN SD CARD SEBAGAI MEDIA PENYIMPANAN

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:

DAHNIAL SYAUQY
NIM. 0510630027-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 23 Desember 2009

DOSEN PENGUJI

Ir. Nanang Sulistyanto
NIP. 19700113 199403 1 002

M. Rif'an, ST.,MT
NIP. 19710301 200012 1 001

Ir. Ponco Siwindarto, M.Eng.Sc.
NIP. 19590304 198903 1 001
Mengetahui
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST., M.Sc
NIP. 19710615 199802 1 003

PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT, karena dengan rahmat, taufik dan hidayah-Nya lah skripsi ini dapat diselesaikan.

Skripsi berjudul "Data *logger* Pengukuran Sinyal Analog Menggunakan *SD card* Sebagai Media Penyimpanan" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Ibu dan Ayah atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Rudy Yuwono, ST., M.Sc selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- M. Aziz Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Ir. M. Julius St, MS selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya dan sekaligus selaku Dosen Pembimbing 1 atas segala ilmu, bimbingan, nasehat, gagasan, ide, saran, motivasi dan bantuan yang telah diberikan,
- DR. Agung Darmawansyah, ST., MT selaku Dosen Pembimbing 2 atas segala bimbingan, nasehat, pengarahan, motivasi, saran dan masukan yang telah diberikan,
- Staf rekording Jurusan Teknik Elektro,
- Teman-teman Laboratorium Elektronika yang bersedia memberikan bantuan ketika dibutuhkan,
- Seluruh teman-teman Streamline angkatan 2005, teman-teman di kelembagaan, senior serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan namanya satu-persatu, terima kasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun.

Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, Desember 2009

Penulis

ABSTRAK

Dahnial Syauqy, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Desember 2009, *Data logger Pengukuran Sinyal Analog Menggunakan SD card Sebagai Media Penyimpanan*, Dosen Pembimbing: Ir. M. Julius St, MS dan DR. Agung Darmawansyah, ST., MT

Data *logger* merupakan perangkat elektronik yang berfungsi untuk merekam data dari suatu perangkat instrumen elektrik. Data yang telah terekam tersebut digunakan untuk analisis lebih lanjut mengenai obyek yang diukur. Data *logger* dirancang untuk mengambil sampel data dan menyimpannya secara periodik. Karena itu, data *logger* harus dapat menyimpan data sesuai dengan ukuran data yang disampel dan seberapa sering data *logger* mengambil data secara periodik.

Sejalan dengan perkembangan teknologi, memori yang murah dengan daya tampung yang besar mudah ditemukan. Salah satu memori tersebut adalah *SD card*. *SD card* menawarkan fitur komunikasi SPI (*Serial Peripheral Interface*) yang mempermudah komunikasi dengan unit pengolah utama berupa mikrokontroler. Basis waktu untuk penentuan kecepatan sampling ditangani oleh IC *Real Time Clock*. Data rekaman yang tersimpan dalam *SD card* disimpan dalam format file teks (*.TXT) dan mengikuti aturan *filesystem* FAT16 agar kemudian dapat dianalisis menggunakan *software* komputer.

Dari hasil pengujian sistem secara keseluruhan didapatkan kesimpulan bahwa sistem dapat bekerja dengan baik dan dapat menyimpan sinyal masukan analog secara periodik ke dalam *SD card* yang selanjutnya dianalisis menggunakan *software* komputer.

Kata Kunci: Data *logger*, sinyal analog, *SD card*, FAT16, file teks

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Data *logger* merupakan salah satu perangkat elektronik yang berfungsi untuk merekam data dari suatu perangkat instrumen elektrik. Data yang telah terekam tersebut digunakan untuk analisis lebih lanjut mengenai obyek yang diukur. Dengan analisis data tersebut, maka dapat diketahui bagaimana karakteristik obyek terukur sehingga dapat dilakukan peningkatan efisiensi pada obyek tersebut. Umumnya data *logger* dirancang untuk mengambil sampel data dan menyimpannya secara periodik. Karena itu, data *logger* harus dapat menyimpan data sesuai dengan ukuran data yang disampel dan seberapa sering data *logger* mengambil data secara periodik. Dengan pertimbangan tersebut, data *logger* memerlukan kapasitas media penyimpan yang relatif berukuran besar. Umumnya, data *logger* langsung terhubung ke *Personal Computer* (PC) yang digunakan untuk menyimpan data sekaligus untuk mengolah data lebih lanjut. Keuntungan sistem PC ini adalah kapasitas penyimpanan yang relatif sangat besar dan kecepatan dalam mengolah data. Namun, sistem ini juga memiliki kelemahan yaitu dalam aplikasi di tempat yang tidak memungkinkan adanya PC. Selain itu, ukuran PC yang relatif besar membuat data *logger* sulit dipindah-pindah untuk pengukuran di banyak tempat.

Dengan pertimbangan tersebut, maka dirancang berbagai data *logger* dengan Mikrokontroler yang berukuran relatif lebih kecil. Media penyimpan data yang umumnya digunakan adalah EEPROM. Kelemahannya adalah daya tampung data yang kecil dan harga yang relatif mahal. Sejalan dengan perkembangan teknologi, berbagai memori yang murah dengan daya tampung yang besar mudah ditemukan. Memori tersebut berupa Flashdisk, MMC dan *SD card*. Media penyimpan berupa flashdisk masih memerlukan chip tambahan yang harganya masih relatif mahal sebagai protokol komunikasi. Oleh karena itu, sebagai solusi dipilih media penyimpan data berupa *SD card*.

Secure Digital Card (*SD card*) merupakan salah satu unit penyimpan yang sering digunakan pada aplikasi peralatan elektronik. *SD card* menggunakan *Serial Peripheral Interface* (SPI) sebagai protokol komunikasi sehingga memungkinkan mikrokontroler untuk mengakses *SD card*. Kapasitas *SD card* yang tersedia di pasar saat ini bermacam-macam. Selain itu, *SD card* dapat diakses menggunakan PC asalkan

data yang tersimpan memenuhi format *filesystem* sistem operasi yang telah ditetapkan. Dengan ukuran *SD card* yang relatif kecil, kapasitas penyimpanan yang relatif besar, kemudahan untuk mengakses menggunakan Mikrokontroler serta kemudahan mengakses data menggunakan PC maka *SD card* dapat diaplikasikan sebagai penyimpan data pada data *logger*.

Umumnya basis waktu yang digunakan dalam prosedur pengambilan sampling data oleh sistem datalogger disediakan oleh mikrokntroler yang juga bekerja sebagai unit pengolah utama. Pembuatan basis waktu menggunakan mikrokontroler tersebut membutuhkan kode pemrograman yang relatif panjang dan rumit. Berawal dari masalah tersebut, maka sistem basis waktu ekternal dan terpisah dengan mikrokontroler sangat dibutuhkan. Sejalan dengan perkembangan teknologi mikroelektronika, IC dengan fungsi *Real Time Clock* (RTC) mempermudah perancangan sistem yang memerlukan penggunaan basis waktu secara *real time*.

1.2 Rumusan Masalah

Dalam perancangan ini rumusan masalah ditekankan pada :

1). Bagaimana merancang dan membuat sistem pewaktuan menggunakan IC RTC DS1307
2). Bagaimana membuat sistem pembacaan waktu dan pembacaan sinyal data input yang direkam dengan menggunakan mikrokontroler R8C/13
3). Bagaimana merancang dan membuat protokol komunikasi data yang memanfaatkan *SD card* sebagai media penyimpanan
4). Bagaimana membuat sistem perekaman data pada *Secure Digital Card* (*SD card*) agar dapat dibaca menggunakan PC

1.3 Ruang Lingkup

Mengacu pada permasalahan yang ada maka tugas akhir ini dibatasi pada:

1). Data masukan yang dapat direkam adalah data sinyal analog DC dengan *range* tegangan antara 0 V sampai tegangan referensi ADC Mikrokontroler sebesar 3,3 V
2). Sensor yang digunakan sebagai sumber penghasil data sinyal analog harus memiliki impedansi kecil antara 0 sampai 825 kΩ sehingga tidak membebani mikrokontroler atau dapat dilewatkan rangkaian buffer terlebih dahulu
3). Data sinyal analog dapat terekam secara efektif hanya jika sinyal analog tersebut memiliki tingkat laju perubahan amplitudo maksimal 0,0275 V/detik

4). Sistem menyediakan pilihan 2 menit, 5 menit, 10 menit dan 30 menit sebagai selang periodik pengambilan dan perekaman

5). *SD card* yang digunakan dalam pembuatan sistem berkapasitas 1 Gb dan diformat dengan *filesystem* FAT16

6). Data disimpan dalam format file teks. Satu file teks berisi rekaman data dalam satu hari penuh

7). Data *logger* hanya digunakan untuk merekam data maksimal selama 1 tahun (365 hari).

8). Data dalam *SD card* dibaca dengan PC dengan menggunakan *card reader* yang telah umum tersedia di pasar

1.4 Tujuan

Tujuan skripsi ini adalah dapat menciptakan sistem perekaman data dalam *SD card* yang mampu menyimpan informasi data dari sensor dengan output sinyal analog secara periodik dan kemudian dapat dianalisis menggunakan PC.

1.5 Sistematika Pembahasan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan sistem.

BAB III Metodologi Penelitian

Berisi tentang metode penelitian dan perencanaan sistem serta pengujian.

BAB IV Perencanaan dan Pembuatan Sistem

Perancangan dan perealisasian sistem yang meliputi spesifikasi, perencanaan diagram blok, prinsip kerja dan realisasi sistem.

BAB V Pengujian Alat

Memuat hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VI Kesimpulan dan Saran

Memuat kesimpulan dan saran-saran untuk pengembangan lebih lanjut dari sistem yang telah dibuat.

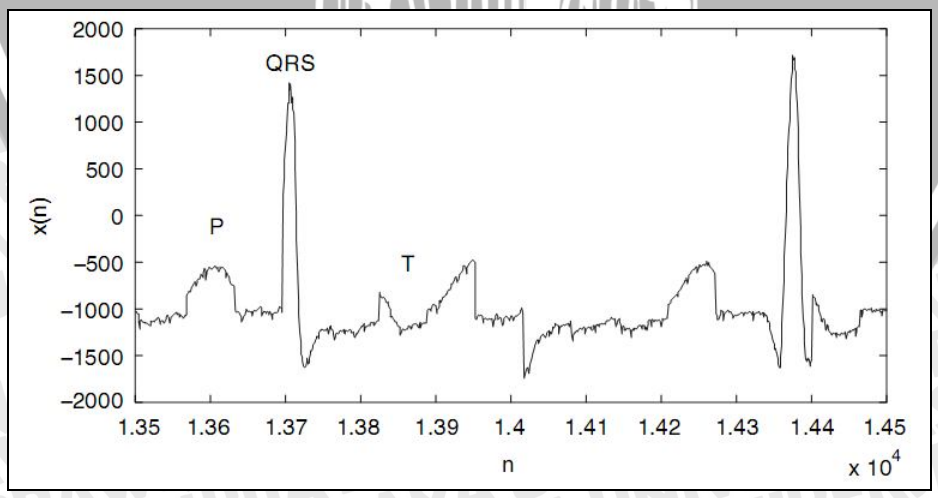
BAB II
TINJAUAN PUSTAKA

Pengetahuan yang dibutuhkan untuk mendukung perencanaan dan realisasi sistem ini meliputi literatur tentang sinyal analog, sistem pewaktuan dengan IC RTC DS1307, mikrokontroler R8C/13 sebagai unit pengolah utama, *SD card* sebagai media penyimpan, *filesystem* FAT16 sebagai struktur dasar pembuatan file dan LCD 16×2 sebagai unit penampil.

2.1 Sinyal Analog

Sinyal adalah nilai atau simbol yang muncul pada urutan yang sama. Sinyal merupakan parameter umum yang ada di bidang keilmuan, teknologi, sosial dan kehidupan. Secara natural, fenomena sinyal yang dapat terukur cenderung bersifat kontinyu. Karena itu, sinyal yang berasal dari alam, contohnya suhu, tekanan, tegangan, aliran, kecepatan dan semacamnya umumnya diukur menggunakan instrumen analog. Untuk mempelajari sinyal tersebut, ilmuwan memodelkan sinyal tersebut dengan persamaan matematis.

Secara matematis, sinyal analog dapat dimodelkan dengan fungsi x:R→R dimana R adalah bilangan real, dan x(t) nilai sinyal pada saat t. Oleh karena itu, secara mudah, sinyal analog dapat diidentifikasi sebagai fungsi dengan variabel real. Gambar 2.1 menunjukkan contoh sinyal analog.

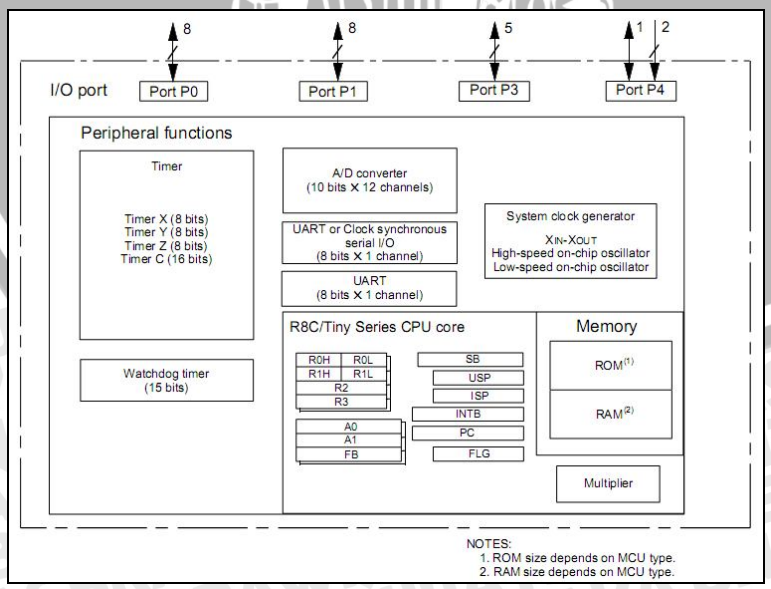


Gambar 2.1. Contoh sinyal analog
Sumber: Ronald L Allen, 2004:14

2.2 Mikrokontroler Renesas R8C/13

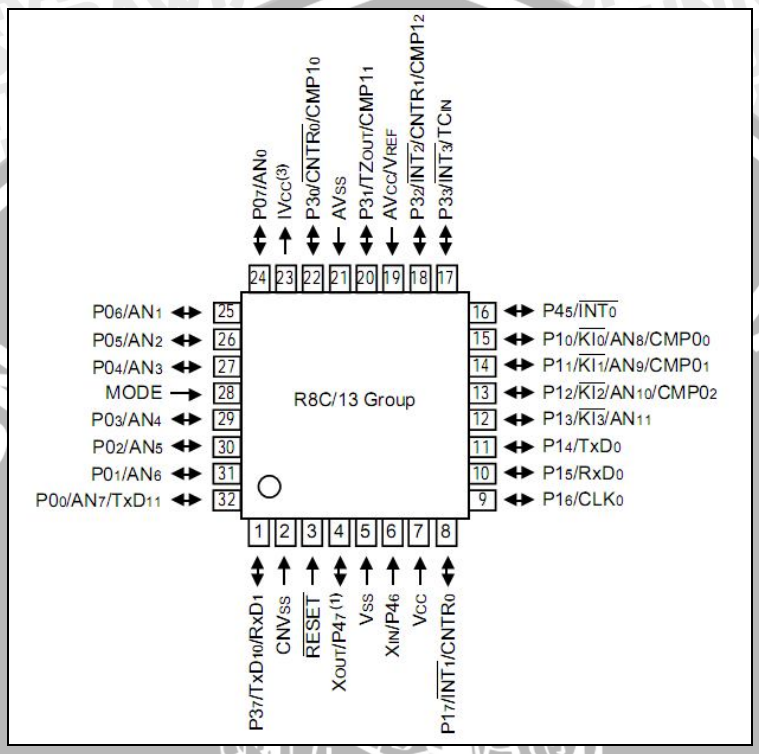
Mikrokontroler R8C/13 merupakan mikrokontoler yang diproduksi oleh Renesas. Mikrokontroler ini memiliki unit pemroses yang berukuran 16-bit sehingga mikrokontroler ini dikategorikan sebagai mikrokontroler 16 bit. Mikrokontroler ini disebut R8C karena memiliki lebar jalur data sebesar 8-bit. Dengan lebar jalur data sebesar 8 bit, maka untuk mengakses data yang berukuran 16-bit, R8C/13 harus mengakses memori sebanyak dua kali. Gambar 2.2 menunjukkan diagram blok mikrokontroler R8C/13. Fitur-fitur yang ada pada R8C/13 adalah sebagai berikut:

- ROM program 16 kB
- RAM 1 KB
- *On-chip oscillator*
- *Watchdog timer*
- Port I/O 22 buah
- Antarmuka serial 2 buah
- *Clock stop detect*
- *Power-on Reset dan Low Voltage Detect*
- ADC dengan resolusi 10-bit sebanyak 12 kanal
- Timer 8-bit 3 buah dengan masing-masing prescaler 8-bit
- Timer 16-bit 1 buah
- Data Flash 2x2 kB dengan fungsi proteksi
- *In circuit programming* dan *debugging*



Gambar 2.2. Diagram blok mikrokontroler R8C/13
Sumber: Renesas, 2005: 3

Secara fisik, mikrokontroler R8C/13 dikemas dalam bentuk *Low Profile Quad Flat Package* yang berukuran 7 mm x 7 mm. Mikrokontroler R8C/13 ini memiliki pin sejumlah 32 buah. Hampir setiap pin mempunyai lebih dari satu fungsi yang hanya bisa dipakai salah satunya dalam satu waktu. Untuk melakukan pengaturan fungsi pin tersebut maka perlu dilakukan *setting* register-register yang bersangkutan. Gambar 2.3 menunjukkan susunan pin R8C/13.



Gambar 2.3. Susunan pin mikrokontroler R8C/13
Sumber: Renesas, 2005: 5

2.2.1 ADC (*Analog to Digital Converter*)

Salah satu fitur yang disediakan oleh R8C/13 adalah 12 kanal ADC dengan waktu konversi yang cukup cepat yang mencapai 2,8 µs. Dengan Adanya fitur ini, maka tidak lagi diperlukan IC ADC eksternal untuk mengkonversi data analog menjadi data digital. Resolusi ADC yang dapat digunakan adalah 8 atau 10 bit. ADC bisa beroperasi dalam 2 mode yaitu *Single Conversion* dan *Free Run Mode*. Pada Mode *Single Conversion*, setiap konversi harus diinisialisasi oleh pengguna. Pada Mode *Free Run*, ADC secara konstan menyampling dan mengupdate Register Data ADC. Mikrokontroler R8C/13 menyediakan 12 kanal ADC yang dapat digunakan. Delapan kanal berada pada port 1, yaitu P_{00} sampai P_{07}. Empat kanal selanjutnya berada pada port 1, yaitu P_{10} sampai P_{13}.

2.2.2 Programmable input output

Programmable input output merupakan salah satu fitur yang disediakan oleh mikrokontroler R8C/13. Mikrokontroler R8C/13 menyediakan 22 buah jalur masukan keluaran yang dapat diatur sesuai dengan kebutuhan. Pengaturan dilakukan untuk menentukan apakah jalur tesebut berfungsi sebagai jalur masukan atau keluaran. Port merupakan kumpulan jalur I/O yang pengaturannya dilakukan dengan menggunakan register yang sama. R8C memiliki empat jenis port, yakni Port 0 yang terdiri atas P_{00}-P_{07}, Port 1 yang terdiri atas P_{10}-P_{17}, Port 3 yang terdiri atas P_{30}-P_{33}, P_{37}, dan Port 4 yang terdiri atas P_{45}. Pada R8C/13 juga terdapat dua jalur tambahan yakni P_{46} dan P_{47} yang dapat digunakan sebagai jalur masukan. Kedua jalur tambahan tersebut dapat digunakan jika *clock* utama yang dipergunakan adalah *clock* internal. Pengaturan fungsi jalur tersebut dilakukan dengan mengatur register delapan bit Pin *Direction* (PD). Tiap bit pada register ini merupakan perwakilan dari tiap jalur pada port yang bersangkutan. Pemberian logika 1 pada register menjadikan jalur berfungsi sebagai jalur keluaran. Sedangkan logika 0 akan mengatur jalur sebagai jalur masukan. Khusus pada register PD_0 terdapat proteksi yang harus dibuka apabila ingin mengganti fungsi input ataupun output. Setelah port diberikan sebuah fungsi, baik sebagai masukan ataupun keluaran, maka selanjutnya port tersebut dapat dibaca ataupun ditulis melalui register port (P).

2.2.3 Antarmuka Serial

Secara umum, mikrokontroler memiliki dua jenis antarmuka serial, yaitu antarmuka serial sinkron dan antarmuka serial asinkron. Dalam antarmuka serial sinkron, sinyal data mengacu pada sebuah sinyal sinkronisasi yang umumnya disebut dengan *clock*. Sedangkan dalam antarmuka serial asinkron, sinyal data mengacu pada laju bit dan format data yang disepakati sebelumnya. Antarmuka serial R8C/13 terdiri atas dua buah yakni UART0 dan UART1. Masing-masing antarmuka serial mempunyai *clock* tersendiri sehingga dapat bekerja secara terpisah. UART0 dapat diatur dalam dua mode, yakni sinkron dan asinkron. UART1 hanya dapat bekerja pada mode asinkron.

Secara *hardware*, mikrokontroler Renesas R8C/13 tidak menyediakan antarmuka *Serial Peripheral Interface* (SPI), namun mode sinkron dapat digunakan untuk menggantikan komunikasi SPI. Pada prinsipnya komunikasi mode sinkron dan SPI adalah sama. Agar sinyal yang dilewatkan dalam komunikasi serial sinkron sama seperti sinyal SPI maka dilakukan pengaturan register-register dalam komunikasi serial sinkron.

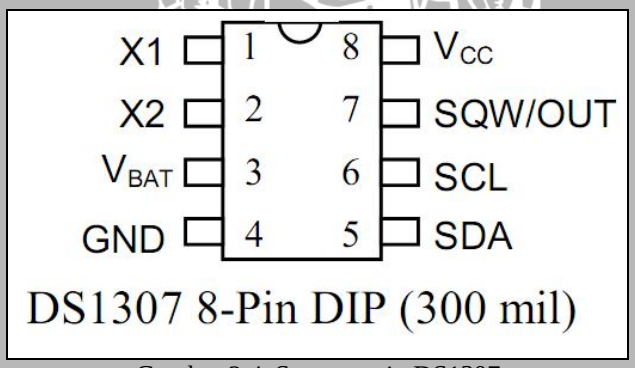
Register utama yang diatur dalam komunikasi serial sinkron adalah register 8 bit U0C0. Register U0C0 bit UFORM digunakan untuk menentukan bit yang dikirim pertama. Bit CKPOL digunakan untuk mengatur data yang akan dikirim pada *clock* tepi naik atau tepi turun. Nilai frekuensi yang digunakan *clock* dapat diatur dengan mengganti nilai yang terdapat pada register U0C0 bit CK0 dan CK1. Data akan dikirim diberikan pada pin TX dan data yang akan diterima pada pin RX. Pin TX dan RX tersebut memiliki fungsi sama dengan pin MOSI dan MISO pada mode SPI.

2.3 RTC DS1307

DS1307 merupakan sebuah IC RTC (*Real Time Clock*) yang dapat merekam dan memberikan informasi waktu lengkap mulai informasi detik, menit, jam, tanggal, bulan hingga tahun. Berbagai fitur yang disediakan chip ini antara lain:

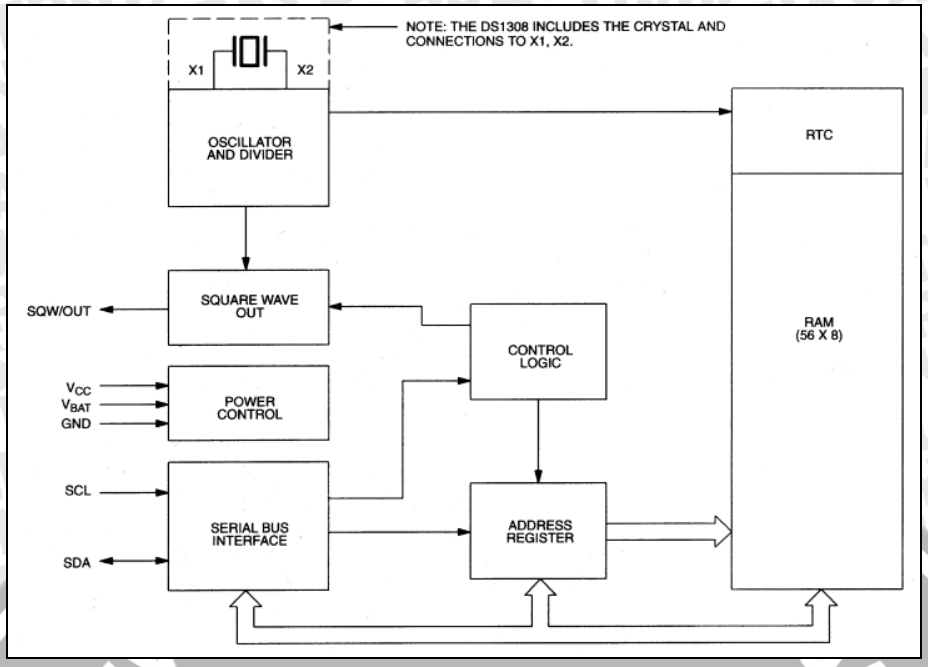
- Perhitungan *real time clock* mulai dari detik menit, jam, tanggal, bulan hingga tahun
- RAM untuk penyimpanan data sebesar 56 byte
- *Interface* I2C yang dikembangkan oleh Philips
- Dapat memberikan output berupa sinyal pulsa yang dapat diprogram
- Mengkonsumsi arus kurang dari 500 nA dalam mode baterai dengan *oscillator* yang tetap berjalan.

Susunan pin IC RTC DS1307 ditunjukkan dalam Gambar 2.4.



Gambar 2.4. Susunan pin DS1307
Sumber: Dallas Semiconductor, 2000: 1

Diagram blok DS1307 ditunjukkan dalam Gambar 2.5.



Gambar 2.5. Diagram blok DS1307
Sumber: Dallas Semiconductor, 2000: 2

X1 dan X2 dihubungkan dengan Xtal berukuran 32,768 kHz. V_{bat} dihubungkan dengan catu daya baterai sebesar 3 V. Jenis komunikasi yang digunakan oleh DS1307 adalah komunikasi serial I2C yang dikembangkan oleh Philips. Komunikasi serial I2C menggunakan dua jalur transmisi, yaitu SCL yang berfungsi sebagai jalur *clock* dan SDA yang berfungsi sebagai jalur data.

Informasi waktu dapat ditulis dan dibaca menggunakan mikrokontroler. Informasi waktu diformat dalam bentuk 2 byte bilangan BCD yang disimpan dalam register-register berbeda. Tabel 2.1 menunjukkan alamat dan isi register dalam DS1307.

Tabel 2.1. Alamat dan isi register DS1307

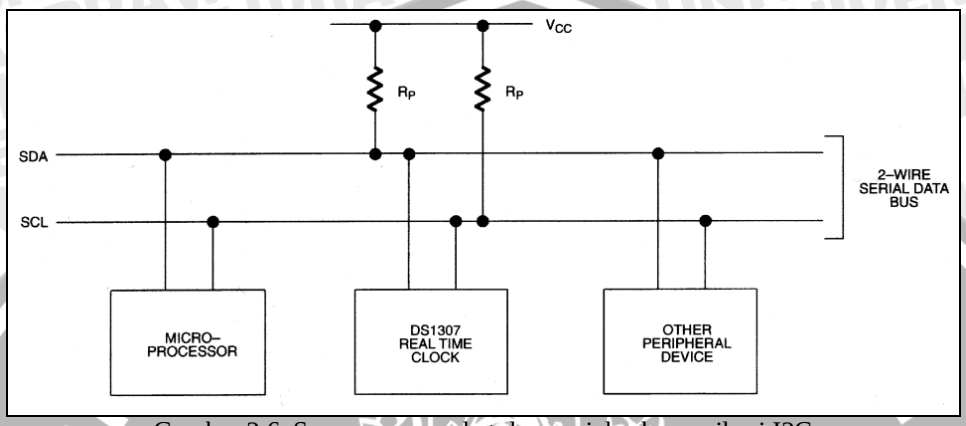
	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	X	10 MINUTES			MINUTES				00–59
	X	12/24	10 HR A/P	10 HR	HOURS				01–12 00–23
	X	X	X	X	X	DAY			1–7
	X	X	10 DATE		DATE				01–28/29 01–30 01–31
	X	X	X	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

Sumber: Dallas Semiconductor, 2000: 4

2.4 Komunikasi I2C

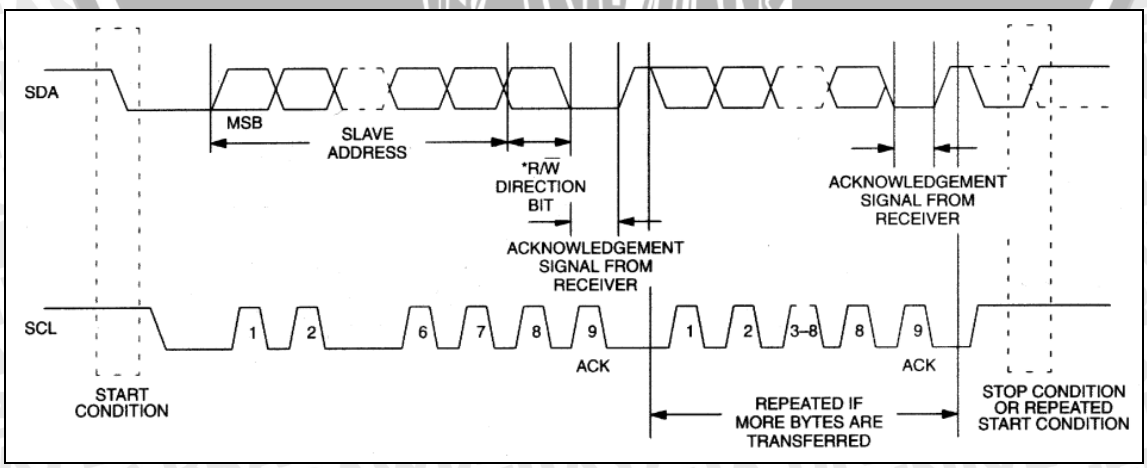
Secara fisik, jalur komunikasi I2C terdiri atas dua kabel. Dua kabel tersebut umumnya dikenal sebagai SDA dan SCL. SDA adalah jalur data baik masuk maupun keluar, sedangkan SCL adalah jalur *clock*.

I2C dapat dihubungkan pada lebih dari satu *slave*. Bentuk umum susunan perangkat yang berkomunikasi dengan I2C ditunjukkan dalam Gambar 2.6.



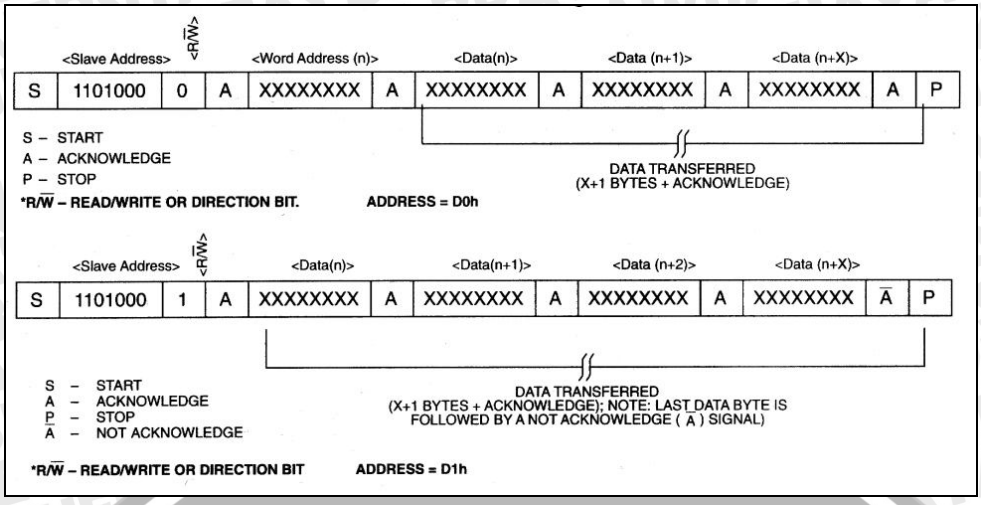
Gambar 2.6. Susunan perangkat dengan jalur komunikasi I2C
Sumber: Dallas Semiconductor, 2000: 5

Urutan komunikasi dalam protokol I2C adalah sebagai berikut. Pertama, mikrokontroler memberikan kondisi start. Kemudian mikrokontroler mengirimkan sinyal alamat perangkat yang akan diakses. Tiap IC *slave* akan membandingkan alamat yang dikirim dengan alamat yang dimiliki. Jika tidak sesuai, maka *slave* akan menunggu hingga master mengirim sinyal stop. Namun jika sesuai, maka *slave* mengirim sinyal *acknowledge*. Saat mikrokontroler menerima sinyal *acknowledge*, maka mikrokontroler dapat mengirimkan data. Setelah selesai, mikrokontroler mengirim sinyal stop. Gambar 2.7 menunjukkan representasi sinyal komunikasi I2C.



Gambar 2.7. Representasi sinyal komunikasi I2C
Sumber: Dallas Semiconductor, 2000: 6

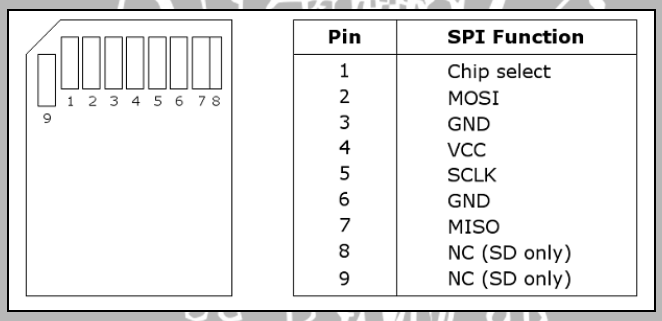
Gambar 2.8 Menunjukkan urutan data dalam komunikasi I2C.



Gambar 2.8. Urutan data dalam komunikasi I2C
Sumber: Dallas Semiconductor, 2000: 7

2.5 *Secure Digital Card (* SD card*)*

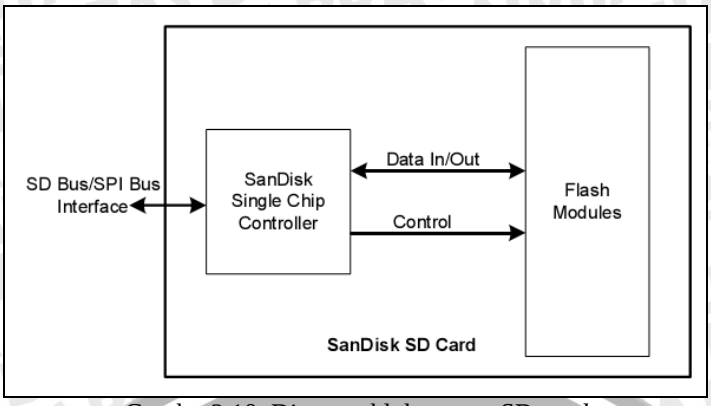
Secure Digital Card merupakan kartu memori yang didesain untuk memberikan kemudahan dan kapasitas yang relatif besar pada perangkat-perangkat elektronik. Komunikasi *SD card* menggunakan sembilan pin yang terdiri atas pin *clock*, perintah, 4 pin data dan 3 pin jalur daya yang dirancang untuk beroperasi pada tegangan rendah. Antarmuka pada *SD card* dirancang untuk mikroprosesor. Gambar 2.9 menunjukkan bentuk fisik dan lokasi pin *SD card.*



Pin	SPI Function
1	Chip select
2	MOSI
3	GND
4	VCC
5	SCLK
6	GND
7	MISO
8	NC (SD only)
9	NC (SD only)

Gambar 2.9. Bentuk fisik dan konfigurasi pin *SD card*
Sumber: Cyan Technology**,** 2008: 5

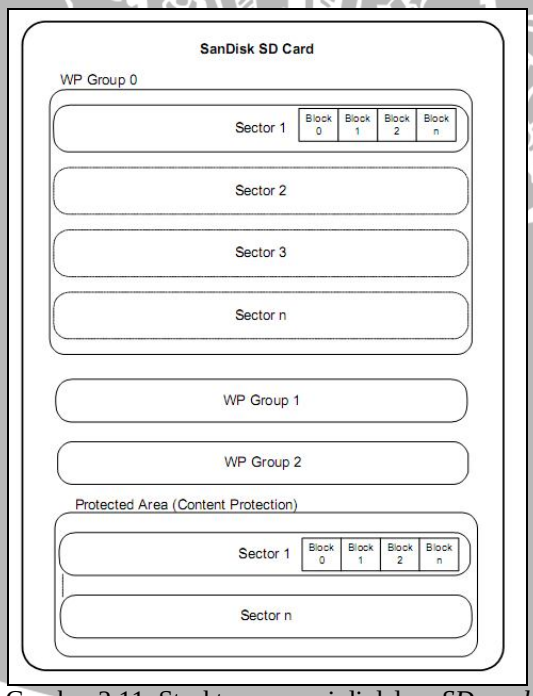
SD card saat ini telah berisi lebih dari 1024 MB memori yang dirancang untuk aplikasi penyimpan data. Dalam *SD card* sudah berisi kontroler yang mengatur protokol antarmuka, algoritma pengamanan dan proteksi hak cipta, algoritma penyimpanan data dan pencarian kembali (*Error Correction Code*/ECC), diagnosa dan penanganan kerusakan, manajemen daya, dan kontrol *clock*. Diagram blok umum *SD card* ditunjukkan dalam Gambar 2.10.



Gambar2.10. Diagram blok umum *SD card*
Sumber: SanDisk, 2003: 1-1

2.5.1 Partisi Memori

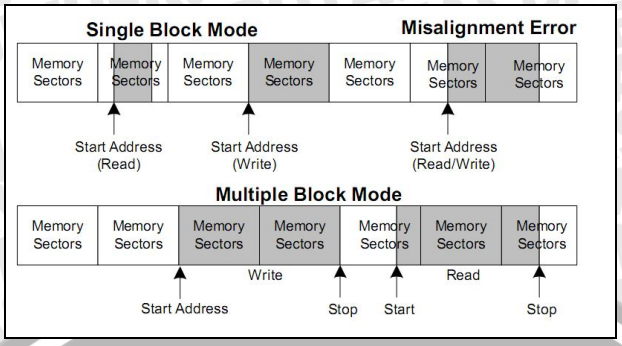
Memori dalam *SD card* dialamati secara byte per byte yang dibagi kedalam beberapa blok, sektor, dan group. Memori di dalam *SD card* terdiri atas blok-blok 512 byte memori yang disebut dengan *sector*. Tiap beberapa *sector* akan dikumpulkan membentuk grup yang disebut *WP Group*. Gambar 2.11 menunjukkan partisi memori dalam *SD card*.



Gambar 2.11. Struktur memori di dalam *SD card*
Sumber : SanDisk, 2007: 1-5

Memori *SD card* dapat diakses blok per blok (*single block mode*) atau beberapa blok sekaligus (*multiple block mode*). Untuk membaca ataupun menulis blok-blok memori tersebut, *master* yang dalam hal ini adalah mikrokontroler, cukup mengirimkan perintah untuk membaca atau menulis beserta alamat memori yang akan diakses. Ini adalah salah satu fitur yang memudahkan karena *master* tidak dilibatkan dalam proses

baca-tulis dalam *SD card.* Gambar 2.12 menunjukkan mode baca tulis berupa *single* dan *multiple block* dalam sebuah *SD card.*



Gambar 2.12 Mode baca/tulis *single* dan *multiple* block
Sumber : SanDisk, 2007: 1-5

2.5.2 Mode Komunikasi

Memori *SD card* dapat diakses dengan menggunakan mode SD bus ataupun mode SPI. Konfigurasi pin untuk tiap mode berbeda-beda. Penjelasan fungsi masing-masing pin untuk tiap-tiap mode komunikasi terdapat dalam Tabel 2.2.

Tabel 2.2. Konfigurasi pin *SD card*

Pin No.	SD Mode			SPI Mode		
	Name	Type	Description	Name	Type	Description
1	CD/DAT	I/O/PP[3]	Card Detect/Data Line [Bit3]	CS	I	Chip Select (neg true)
2	CMD	PP	Command/Response	DI	I	Data In
3	V_{SS1}	S	Supply voltage ground	VSS	S	Supply voltage ground
4	V_{DD}	S	Supply voltage	VDD	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	V_{SS2}	S	Supply voltage ground	VSS2	S	Supply voltage ground
7	DAT0	I/O/PP	Data Line [Bit0]	DO	O/PP	Data Out
8	DAT1	I/O/PP	Data Line [Bit1]	RSV		
9	DAT2	I/O/PP	Data Line [Bit2]	RSV		

Sumber: Transcend, 2009:1

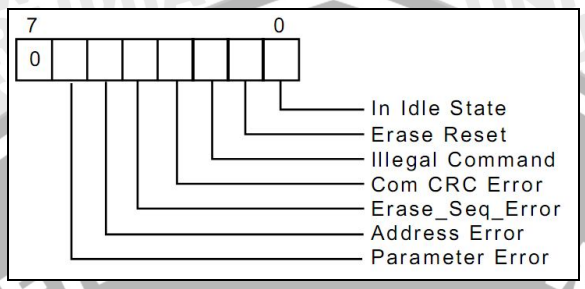
Mode SPI adalah protokol komunikasi sekunder yang ada dalam *SD card*. Tiap Data dibangun dari 8 bit (1 byte) dan selalu berdasar pada sinyal *clock*. Data SPI di dalam *SD card* dibangun dari *command, responsse* dan *data tokens.* Semua komunikasi antara *master dan SD card* diatur oleh *master* yang umumnya berupa mikrokontroler atau mikroprosesor. Panjang paket perintah (*command frame*) dari master ke *SD card* adalah sebesar 6 byte. Byte CRC bersifat opsional dalam mode SPI, tapi harus diisi untuk membentuk paket perintah yang selalu 6 byte. Tabel 2.3 menunjukkan tabel *command frame*.

Tabel 2.3. *Command frame SD card*

Byte 1				Bytes 2—5		Byte 6	
7	6	5	0	31	0	7	0
0	1	Command		Command Argument		CRC	1

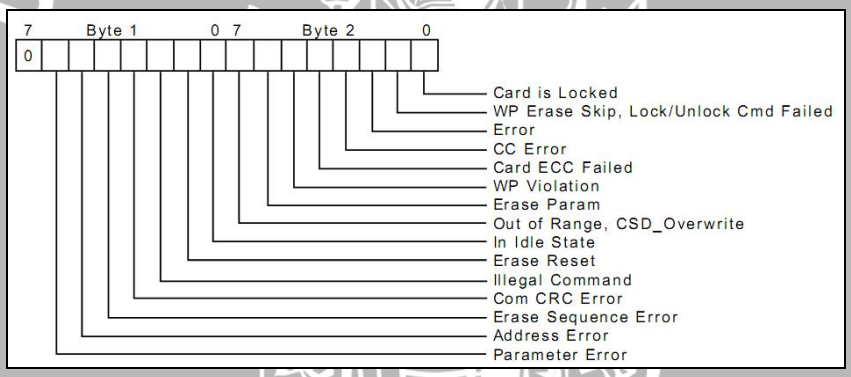
Sumber : SanDisk, 2003: 5-8

Saat *master* mengirimkan perintah ke *SD card*, maka akan ada respons dari *SD card* yang dikirim balik ke *master*. Jenis respons yang dikirim *SD card* tergantung dari jenis perintah yang diterima. Waktu respons perintah (*command responsse time – NCR*) adalah selama 1 sampai 8 byte. Ada tiga jenis respons terhadap perintah yang dikirim ke *SD card*, yaitu R1, R2, dan R3. Bit-bit dalam respons R1 ditunjukkan dalam Gambar 2.13.



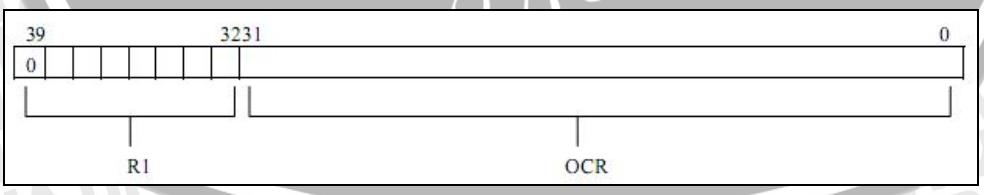
Gambar 2.13 Respons R1
Sumber : SanDisk, 2003: 5-13

Bit-bit dalam respons R2 ditunjukkan dalam Gambar 2.14



Gambar 2.14 Respons R2
Sumber : SanDisk, 2003: 5-14

Bit-bit dalam respons R3 ditunjukkan dalam Gambar 2.15



Gambar 2.15 Respons R3
Sumber : SanDisk, 2003: 5-14

Setiap perintah di dalam komunikasi SPI *SD card* diekpresikan dalam bentuk CMD<x>, dengan <x> adalah nomor indeks perintah yang bernilai 0 sampai 63. Tabel 2.4 menunjukkan perintah-perintah yang sering dipakai untuk operasi baca/tulis dan inisialisasi *SD card* beserta respons yang diberikan oleh *SD card* kepada *master*.

Tabel 2.4 Perintah-perintah umum dalam *SD card*

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	None	R1	GO_IDLE_STATE	Resets the SD Card
CMD1	Yes	None	R1	SEND_OP_COND	Activates the card's initialization process.
CMD2	No				
CMD3	No				
CMD4	No				
CMD5	Reserved				
CMD6	Reserved				
CMD7	No				
CMD8	Reserved				
CMD9	Yes	None	R1	SEND_CSD	Asks the selected card to send its card-specific data (CSD).
CMD10	Yes	None	R1	SEND_CID	Asks the selected card to send its card identification (CID).
CMD11	No				
CMD12	Yes	None	R1b	STOP _TRANSMISSION	Forces the card to stop transmission during a multiple block read operation.
CMD13	Yes	None	R2	SEND_STATUS	Asks the selected card to send its status register.
CMD14	No				
CMD15	No				
CMD16	Yes	[31:0] block length	R1	SET_BLOCKLEN	Selects a block length (in bytes) for all following block commands (read & write).[1]
CMD17	Yes	[31:0] data address	R1	READ_SINGLE _BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[2]
CMD18	Yes	[31:0] data address	R1	READ_MULTIPLE _BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_ TRANSMISSION command.
CMD19	Reserved				
CMD20	No				
CMD21 ... CMD23	Reserved				
CMD24	Yes	[31:0] data address	R1[3]	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[4]
CMD25	Yes	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a stop transmission token is sent (instead of 'start block').
CMD26	No				
CMD27	Yes	None	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28[1]	Yes	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29[4]	Yes	[31:0] data address	R1b	CLR_WRITE_PROT	If the card has write protection features, this command clears the write protection bit of the addressed group.
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_ PROT	If the card has write protection features, this command asks the card to send the status of the write protection bits.[2]
CMD31	Reserved				
CMD32	Yes	[31:0] data address	R1	ERASE_WR_BLK_START _ADDR	Sets the address of the first write block to be erased.
CMD33	Yes	[31:0] data address	R1	ERASE_WR_BLK_END _ADDR	Sets the address of the last write block in a continuous range to be erased.
CMD34 .... CMD37	Reserved				
CMD38	Yes	[31:0] don't care*	R1b	ERASE	Erases all previously selected write blocks.
CMD39	No				
CMD40	No				
CMD41 ... CMD54	Reserved				
CMD55	Yes	[31:0] stuff bits	R1	APP_CMD	Notifies the card that the next command is an application specific command rather than a standard command.
CMD56	Yes	[31:0] stuff bits [0]: RD/WR[3]	R1	GEN_CMD	Used either to transfer a Data Block to the card or to get a Data Block from the card for general purpose/application specific commands. The size of the Data Block is defined with SET_BLOCK_LEN command.
CMD57	Reserved				
CMD58	Yes	None	R3	READ_OCR	Reads the OCR register of a card.
CMD59	Yes	[31:1] don't care* [0:0] CRC option	R1	CRC_ON_OFF	Turns the CRC option on or off. A '1' in the CRC option bit will turn the option on, a '0' will turn it off.
CMD60-63	No				

Sumber : SanDisk, 2003:5-10

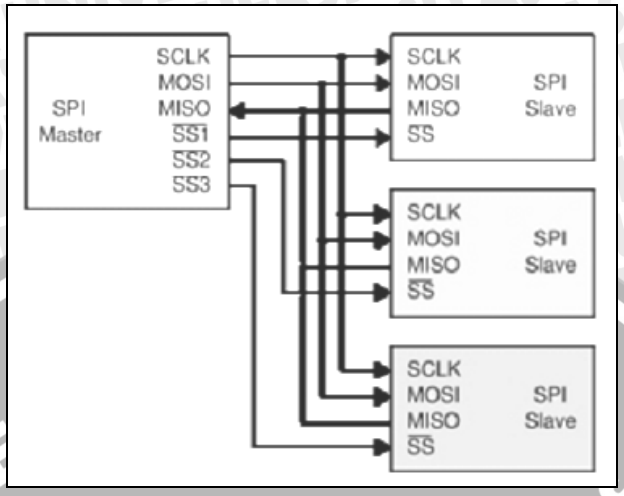
Saat *SD card* pertama kali diaktifkan, mode komunikasi yang digunakan adalah mode SD bus. Mode komunikasi dapat diubah menjadi SPI dengan cara memberikan CS logika *low* saat penerimaan *command reset* (CMD0). Ketika pertama kali dihidupkan, tunggu beberapa milisekon hingga SDC memasuki mode operasi aslinya (protokol SD bus). Setelah itu set DI dan CS dengan logika *high* dan berikan sedikitnya 74 pulsa *clock* pada pin SCLK. Kirimkan perintah CMD0 dengan sinyal CS *low* untuk me-reset *SD card*. Pada saat CMD0 diterima, *SD card* memasuki kondisi *idle* dan respons R1 adalah 0x01 (bit *In-Idle-State* bernilai 1). Dalam keadaan *idle, SD card* hanya dapat menerima perintah CMD0, CMD1, dan CMD58. Jika pada saat *idle SD card* menerima perintah CMD1, maka inisialisasi akan dimulai. Untuk mengetahui akhir inisialisasi, *master* harus melakukan cek terhadap respons. Inisialisasi sudah selesai jika respons R1 kembali bernilai 0x00 (bit *In-Idle-State* bernilai 0). Proses inisialisasi memakan waktu beberapa ratus milisekon. Setelah inisialisasi, operasi baca/tulis dalam mode SPI dapat dilakukan.

2.6 Komunikasi SPI (*Serial Peripheral Interface*)

SPI (*serial peripheral interface bus*) merupakan salah satu metode pengiriman data dari suatu devais ke devais lainnya. Metode ini merupakan metode yang bekerja pada metode *full duplex* dan merupakan standar sinkronisasi serial data link yang dikembangkan oleh Motorola. Dalam komunikasi SPI, devais dibagi menjadi dua bagian yaitu *master* dan *slave*. *Master* bertindak sebagai devais utama yang menginisiasi pengiriman data. Dalam aplikasinya, sebuah *master* dapat digunakan untuk mengatur pengiriman data dari atau ke beberapa *slave* sekaligus. SPI terkadang disebut juga dengan *four wire* serial bus untuk membedakannya dengan bus serial tiga, dua, dan satu kabel. Komunikasi serial antara *master* dan *slave* dalam SPI diatur melalui 4 buah pin yang terdiri atas SCLK, MOSI, MISO, dan SS.

SCLK (*serial clock*) atau terkadang disebut SCK merupakan data biner yang keluar dari *master* ke *slave* yang berfungsi sebagai *clock* dengan frekuensi tertentu. MOSI (*master out slave input*) merupakan pin yang berfungsi sebagai jalur data pada saat data keluar dari *master* dan masuk ke dalam *slave*. MISO (*master input slave output*) merupakan pin yang berfungsi sebagai jalur data yang keluar dari *slave* dan masuk ke dalam *master*. SS (*slave select*) merupakan pin yang berfungsi untuk mengaktifkan *slave* sehingga pengiriman data hanya dapat dilakukan jika *slave* dalam keadaan aktif (*active low*). Pin SCLK, MOSI, dan SS merupakan pin

dengan arah pengiriman data dari *master* ke *slave*. Sebaliknya, MISO mempunyai arah komunikasi data dari *slave* ke *master*. Gambar 2.16 menunjukkan rangkaian dasar komunikasi SPI antara *master* dan beberapa *slave*.



Gambar 2.16 Konfigurasi rangkaian SPI
Sumber : Amin Mutohar, 2008: 4

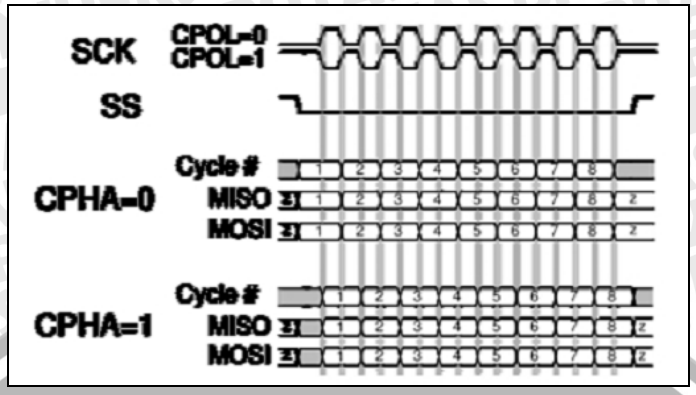
Komunikasi data SPI dimulai pada saat *master* mengirimkan *clock* melalui SCK dengan frekuensi lebih kecil atau sama dengan frekuensi maksimum pada *slave*. Kemudian, *master* memberi logika nol pada SS untuk mengaktifkan *slave* sehingga pengiriman data (berupa siklus *clock*) siap untuk dilakukan. Pada saat siklus *clock* terjadi transmisi data *full duplex* terjadi dengan dua keadaan sebagai berikut:

- *Master* mengirim sebuah bit pada jalur MOSI, *slave* membacanya pada jalur yang sama.
- *Slave* mengirim sebuah bit pada jalur MISO, *master* membacanya pada jalur yang sama.

Transmisi dapat menghasilkan beberapa siklus *clock*. Jika tidak ada data yang dikirim lagi maka master menghentikan *clock* tersebut dan kemudian menon-aktifkan *slave*.

Diagram pewaktuan (*timing diagram*) SPI dimulai pada saat SS diaktifkan (*low*). Pada saat tersebut siklus *clock* (*cycle #*) dimulai. Saat SS aktif, MISO/MOSI mulai mengirimkan data mulai dari MSB (*most significant bit*) data tersebut. Pada saat *clock* berubah maka proses pengiriman data dilanjutkan pada bit yang lebih rendah. Proses tersebut berlangsung sampai pengiriman data selesai dengan mengirimkan bit LSB (*least significant bit*) dan siklus *clock* berakhir serta SS kembali dinon-aktifkan (*high*). Pada saat ini biasanya *slave* mengirimkan *interrupt* ke *master* yang mengindikasikan bahwa pengiriman data telah selesai dan siap untuk

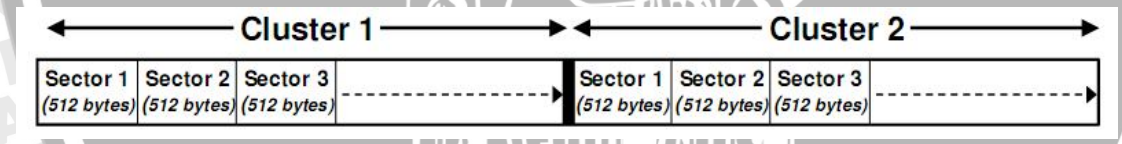
melakukan pengiriman data selanjutnya. Gambar 2.17 menunjukkan timing diagram komunikasi SPI.



Gambar 2.17 Timing diagram komunikasi SPI
Sumber : Amin Mutohar, 2008: 3

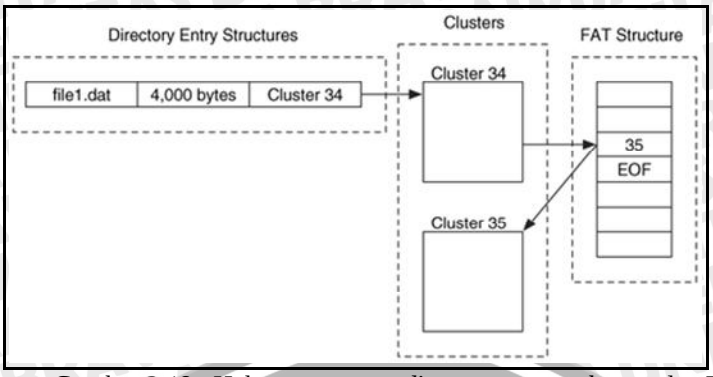
2.7 Sistem File FAT

FAT (*File Allocation Table*) merupakan salah satu *filesystem* paling sederhana yang umum dipakai dalam banyak sistem operasi. Di dalam *filesystem* FAT, setiap file atau *directory* dialokasikan dalam sebuah struktur data yang disebut *directory entry* yang berisi informasi mengenai nama file, ukuran, alamat awal dan beberapa informasi lain. Setiap file disimpan dalam satuan data yang disebut *cluster*. *Cluster* adalah kumpulan dari beberapa *sector*. Sedangkan setiap *sector* berisi 512 byte data. Gambar 2.18 menunjukkan struktur pengelompokan data.



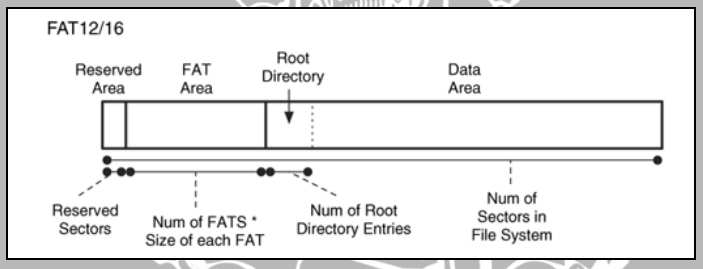
Gambar 2.18 struktur pengelompokan data.

Jika sebuah file dialokasikan lebih dari satu *cluster* maka alamat *cluster* selanjutnya dapat ditemukan dalam suatu struktur yang disebut FAT. Struktur FAT digunakan untuk mengetahui lokasi *cluster* selanjutnya dari sebuah file dan juga status *cluster* apakah sudah terpakai atau belum. Struktur FAT dapat memiliki ukuran *entry* yang berbeda-beda. Perbedaan ukuran tersebut menyebabkan adanya beberapa jenis *filesystem* FAT. Saat ini dikenal ada tiga macam versi FAT, yaitu FAT12, FAT16, dan FAT32. Gambar 2.19 menunjukkan hubungan antara *directory entry*, *cluster*, dan FAT.



Gambar 2.19. Hubungan antara *directory entry*, *cluster*, dan FAT
Sumber : Brian Carrier, 2005

Secara fisik, FAT memiliki tiga bagian umum seperti yang ditunjukkan dalam Gambar 2.20. Bagian pertama adalah *reserved area* yang berisi data *reserved* dalam filesystem. Ukuran *reserved area* tersebut ditentukan di dalam *boot sector*. Bagian kedua adalah FAT area yang berisi struktur FAT utama dan *back-up*nya. Bagian ketiga adalah data area yang berisi cluster dan dipergunakan untuk menyimpan file dan direktori.



Gambar 2.20. Struktur fisik FAT
Sumber : Brian Carrier, 2005

Directory entry f atas 32 byte data yang berisi nama file, ukuran, alamat awal dari isi file, dan informasi tanggal dan waktu file dibuat, seperti ditunjukkan dalam Tabel 2.5.

Tabel 2.5. Isi *directory entry*

Byte	**Keterangan**
0–0	Karakter pertama nama file dalam ASCII dan status alokasi (0xE5 atau 0x00 jika belum dialokasikan/terpakai)
1–10	Karakter 2 sampai 11 nama file dalam ASCII
11–11	Attribut file (*read only*, *write only*, *archive*, dsb.)
12–12	Reserved
13–13	Waktu file dibuat (kelipatan 10 sekon)
14–15	Waktu file dibuat (jam, menit, detik)
16–17	Hari file dibuat
18–19	Hari file diakses
20–21	Alamat cluster pertama (0 untuk FAT12 and FAT16), 2 byte atas

Byte	Keterangan
22–23	Waktu file ditulis (jam, menit, detik)
24–25	Hari file ditulis
26–27	Alamat cluster pertama, 2 byte bawah
28–31	Ukuran file (dalam byte)

Sumber: Brian Carrier, 2005

Susunan memori sebuah *hard-disk* atau *SD card* terdiri atas MBR (*Master Boot Record*) dan partisi-partisi nya. MBR selalu berada di bagian paling awal dari sebuah struktur memory, yakni *sector* 0. MBR adalah sebuah set kode awal yang akan terbaca oleh komputer. Tujuan utama sebuah MBR adalah untuk membaca struktur *filesystem* dari *memory* yang diakses. MBR umumnya berukuran 512 byte serta mengandung informasi tabel partisi. Tabel 2.6 menunjukkan isi MBR.

Tabel 2.6 *Master Boot Record*

Byte Position	Length (bytes)	Entry Description	Value/Range
0x0	446	Consistency check routine	---
0x1be	16	Partition Table entry	—
0x1ce	16	Partition Table entry	—
0x1de	16	Partition Table entry	—
0x1ee	16	Partition Table entry	—
0x1fe	1	Signature	0x55
0x1ff	1	Signature	0xaa

Sumber: SanDisk, 2005: 3-21

Tabel partisi berisi informasi tentang lokasi dan ukuran setiap partisi di dalam *memory*. Tabel 2.7 menunjukkan isi *partition table*.

Tabel 2.7. *Partition table*

Byte Position	Length (bytes)	Entry Description	Value/Range
0x0	1	Boot descriptor	0x00 (Non-bootable device) 0x80 (Bootable device)
0x1	3	First partition sector	Address of First Sector
0x4	1	File system descriptor	0 = Empty 1 = DOS 12-bit FAT < 16 MB 4 = DOS 16-bit FAT < 32 MB 5 = Extended DOS 6 = DOS 16-bit FAT >=32 MB 0x10 – 0xFF = Free for other File System
0x5	3	Last partition sector	Address of last sector
0x8	4	First sector positition relative to beginning of device	Number of first sector (Linear Address)
0xC	4	Number of sectors in partition	Between one and max. amount of sectors on device

Sumber: SanDisk, 2005: 3-22

Dari tabel partisi tersebut, dapat diketahui alamat sektor pertama suatu partisi (*Address of First Sector*). Sektor pertama tersebut dinamakan *Volume Boot Record*

(VBR), yang berisi informasi spesifik mengenai partisi yang bersangkutan. Parameter-parameter yang penting dalam VBR antara lain adalah ukuran *bytes per sector*, *sectors per cluster*, *reserved sectors*, jumlah FAT, dan jumlah *entry* dalam *root directory*. Tabel 2.8 menunjukkan VBR dan informasi partisi yang diberikan.

Tabel 2.8. *Volume Boot Record*

Byte Position	Length (bytes)	Entry Description	Value/Range
0x0	3	Jump command	0xeb 0xXX 0x90
0x3	8	OEM name	XXX
0xb	2	Bytes/sector	512
0xd	1	Sectors/cluster	XXX (range: 1-64)
0xe	2	Reserved Sectors (Number of reserved sectors at the beginning of the media including the boot sector.)	1
0x10	1	Number of FATs	2
0x11	2	Number of root directory entries	512
0x13	2	Number of sectors on media	XXX (Depends on card capacity, if the media has more than 65535 sectors, this field is zero and the 'number of total sectors' is set.)
0x15	1	Media descriptor	0xf8 (hard disk)
0x16	2	Sectors/FAT	XXX
0x18	2	Sectors/track	32 (no meaning)
0x1a	2	Number of heads	2 (no meaning)
0x1c	4	Number of hidden sectors	0
0x20	4	Number of total sectors	XXX (depends on capacity)
0x24	1	Drive number	0x80
0x25	1	Reserved	0
0x26	1	Extended boot signature	0x29
0x27	4	Volume ID or serial number	XXX
0x2b	11	Volume label	XXX (ASCII characters padded with blanks if less than 11 characters.)
0x36	8	File system type	XXX (ASCII characters identifying the file system type FAT12 or FAT16.)
0x3e	448	Load program code	XXX
0x1fe	1	Signature	0x55
0x1ff	1	Signature	0xaa

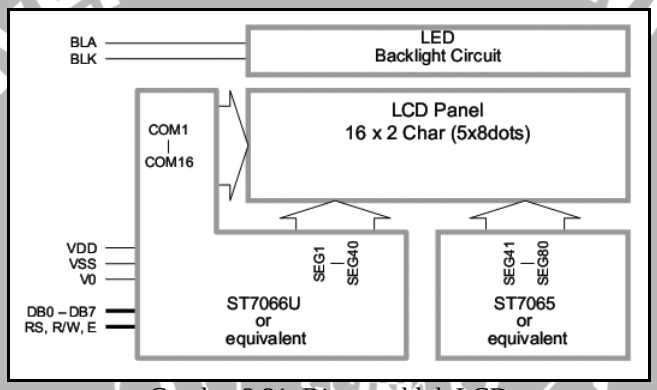
Sumber: SanDisk, 2005: 3-22

2.6 Modul *Liquid Cristal Display* (LCD)

Dalam sistem ini, LCD digunakan sebagai penampil untuk memudahkan pengguna dalam pengoperasian alat. Fitur-fitur modul LCD ini adalah sebagai berikut.

- Dapat menampilkan 16 karakter×2 baris
- Setiap huruf disusun 5×8 titik
- *Character Code* ROM
- *Character Generator* RAM (CGRAM) dan *Display Data* RAM (DDRAM)
- Dua mode antarmuka, yaitu 4 bit dan 8 bit.

Mode antarmuka 4 bit biasa digunakan untuk menghemat penggunaan pin mikrokontroler. Gambar 2.21 menunjukkan diagram blok LCD. Sedangkan fungsi pin LCD ditunjukkan dalam Tabel 2.9.



Gambar 2.21. Diagram blok LCD
Sumber : Hitachi, 2006: 3

Tabel 2.9. Fungsi pin LCD

No. pin	Nama pin	I/O	Keterangan
1	VSS	Catu daya	Ground (0 volt)
2	VDD	Catu daya	Positif catu daya (5volt)
3	V0	Catu daya	Tegangan referensi kontras LCD
4	RS	Input	*Register Select* RS=HIGH: kirim data *display* RS=LOW: kirim data instruksi
5	R/W	Input	*Read/Write Control Bus* R/W=HIGH: mode baca R/W=LOW: mode tulis
6	E	Input	*Data enable*
7	DB0	I/O	*Bi-directional Tri-state Data Bus* Mode 8 bit: gunakan DB0 – DB7 Mode 4 bit: gunakan DB4 – DB7
:	:		
14	DB7		
15	BLA	Catu daya	Positif catu daya *backlight*
16	BLK	Catu daya	Negatif catu daya *backlight*

Sumber: Hitachi, 2006: 4

BAB III
METODOLOGI PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat menampilkan unjuk kerja sesuai yang direncanakan dengan mengacu pada rumusan masalah. Pemilihan komponen dilakukan berdasarkan perencanaan dan disesuaikan dengan komponen yang mudah dijumpai di pasaran. Untuk merealisasikan sistem yang telah dirancang, langkah-langkah yang dilakukan adalah sebagai berikut:

1. Studi Literatur
2. Perancangan Alat
3. Pembuatan Alat
4. Pengujian Sistem
5. Pengambilan Kesimpulan dan Saran

3.1 Studi Literatur

Studi literatur mengacu pada spesifikasi yang dibuat untuk memahami komponen pendukung yang diperlukan guna merealisasikan alat. Studi literatur yang dilakukan meliputi mikrokontroler R8C/13, RTC DS1307, modul LCD, *filesystem* FAT16 dan format file teks serta *Secure Digital Card* (*SD card*).

a. Studi tentang mikrokontroler R8C/13
   - *Hardware* R8C/13
   - Pemrograman sistem mikrokontroler R8C/13 dengan bahasa C
   - Fasilitas internal mikrokontroler R8C/13:
     1. Struktur dan operasi port-port
     2. *Analog to Digital Converter* (ADC)
     3. Komunikasi serial

b. Studi tentang RTC DS1307
   - *Datasheet* dan *hardware* DS1307
   - Komunikasi I2C dan pemrograman register DS1307

c. Studi tentang LCD 2×16
   - Rangkaian dan datasheet LCD 2×16

d. Studi tentang filesystem FAT16
   - Struktur dan organisasi file dalam *filesystem* FAT16

e. Studi tentang *SD card*

-*Datasheet* dan *hardware SD card*

-Komunikasi SPI dan pemrograman *SD card*

3.2 Perancangan Alat

Pada tahap perancangan alat, dibuat suatu blok diagram fungsional dari rangkaian yang direncanakan. Perancangan rangkaian dilakukan pada tiap-tiap blok untuk mempermudah perancangan serta penentuan nilai komponen yang digunakan. Perancangan perangkat lunak dilakukan dengan membuat diagram alir untuk program utama dan sub program. Secara garis besar perancangan alat dilakukan dalam tahap berikut:

1. Penentuan spesifikasi alat.
2. Pembuatan diagram blok sistem keseluruhan.
3. Perancangan perangkat keras yang terdiri atas rangkaian push button, LCD, rangkaian RTC, Rangkaian *bidirectional level shifter* 3,3 V- 5 V dan antarmuka *SD card*.
4. Perancangan perangkat lunak yang terdiri atas software mikrokontroler dan software komputer.

3.3 Pembuatan Alat

Pembuatan alat dalam skripsi ini meliputi :

a. Pembuatan perangkat keras sistem dengan menggunakan komponen elektronika yang telah direncanakan.

b. Pembuatan perangkat lunak mikrokontroler sesuai dengan diagram alir yang telah direncanakan.

3.4 Pengujian Sistem

Pengujian sistem dilakukan untuk mengetahui kinerja alat apakah sesuai dengan yang dirancang. Pengujian dilakukan pada masing-masing blok dan kemudian secara keseluruhan sistem. Secara garis besar pengujian perblok adalah sebagai berikut:

a. Pengujian ADC dan rangkaian input *push button*

Tujuan pengujian ini adalah untuk mengetahui apakah mikrokontroler beserta fasilitas *programmable input output* dan *ADC* bekerja dengan baik. Pengujian dilakukan dengan memberikan tegangan yang diatur oleh potensiometer pada salah satu pin ADC dan kemudian Mikrokontroler diisi

program sederhana guna membaca data ADC. Pengujian rangkaian *push button* dilakukan dengan menghubungkan *push button* dengan pin mikrokontroler. Data yang terbaca dapat diamati melalui *software* HEW yang disediakan oleh Renesas. Dari pengujian akan diamati apakah mikrokontroler dapat menerima masukan ADC dan *push button*.

b. Pengujian *SD card*

Pengujian *Secure Digital Card* dilakukan untuk mengetahui apakah *SD card* dapat dibaca dan ditulis dengan benar. Pengujian dilakukan dengan menghubungkan *SD card* dan mikrokontroler kemudian dilakukan penulisan dan pembacaan data sederhana. Data pada *SD card* akan dicocokkan dengan data pada mikrokontroler. Pengamatan data dilakukan dengan *software* HEW dan WinHex.

c. Pengujian Rangkaian Level shifter 3,3 V-5 V

Pengujian ini dilakukan untuk mengetahui apakah Rangkaian level shifter dapat berfungsi dengan benar. Pengujian dilakukan dengan memberikan sinyal diskrit baik dengan amplitudo 3,3 V maupun 5 V. Keluaran rangkaian level shifter kemudian diamati dengan osiloskop.

d. Pengujian IC RTC DS1307

Pengujian IC RTC bertujuan untuk mengetahui apakah IC RTC dapat memberikan informasi waktu dengan benar. Pengujian dilakukan dengan menghubungkan IC RTC dan mikrokontroler kemudian data waktu dituliskan ke dalam IC RTC dan selanjutnya dibaca kembali oleh mikrokontroler.

e. Pengujian Modul LCD

Pengujian modul LCD dilakukan untuk mengetahui apakah LCD dapat menampilkan karakter dengan benar. Pengujian dilakukan dengan menghubungkan LCD dan mikrokontroler kemudian menampilkan beberapa karakter.

f. Pengujian sistem secara keseluruhan

Pengujian sistem keseluruhan dilakukan untuk menguji apakah sistem data *logger* dapat bekerja dengan benar sesuai dengan spesifikasi yang dirancang yakni merekam data ADC secara periodik dengan selang waktu tertentu ke dalam *SD card* dalam format file teks yang dapat dibaca melalui komputer. Pengujian dilakukan dengan memberikan masukan ADC berupa output analog dari sensor.

Karena jumlah kanal ADC yang digunakan adalah dua kanal, maka sensor yang digunakan adalah dua sensor yang dihubungkan pada masing-masing kanal ADC. Kemudian setelah jangka waktu tertentu, data yang telah terekam dalam *SD card* dibaca menggunakan komputer. LCD dapat digunakan untuk mengetahui data ADC yang terbaca pada waktu tertentu.

3.5 Pengambilan Kesimpulan dan Saran

Tahap berikutnya dalam penelitian ini adalah pengambilan kesimpulan dari alat yang dibuat. Pengambilan kesimpulan ini didasarkan pada kesesuaian antara teori dan praktik. Tahap terakhir penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang mungkin terjadi, kemungkinan pengembangan, serta penyempurnaan pembuatan alat di masa mendatang.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Bab ini menjelaskan tentang perancangan sekaligus pembuatan sistem data *logger* pengukuran analog menggunakan *SD card* sebagai media penyimpanan. Alat dirancang untuk dapat merekam data sinyal analog ke dalam *SD card* dengan sampling pengambilan data berdasarkan masukan *push button*. Data hasil rekaman disimpan dalam format file teks yang selanjutnya dianalisis dengan *software* komputer untuk menampilkan data historik berupa tabel dan grafik. Perancangan sistem ini meliputi perancangan perangkat keras (*hardware*) dan perangkat lunak (*software*). Perancangan perangkat keras (*hardware*) meliputi rangkaian *push button*, rangkaian IC RTC (*Real Time Clock*), rangkaian *bidirectional level shifter*, LCD dan antarmuka SPI *SD card*. Sedangkan perangkat lunak (*software*) meliputi program untuk mikrokontroler Renesas R8C/13 dengan menggunakan bahasa C serta program yang digunakan untuk membaca data hasil rekaman melalui komputer dengan menggunakan bahasa pemrograman Delphi.

4.1 Penentuan Spesifikasi Alat

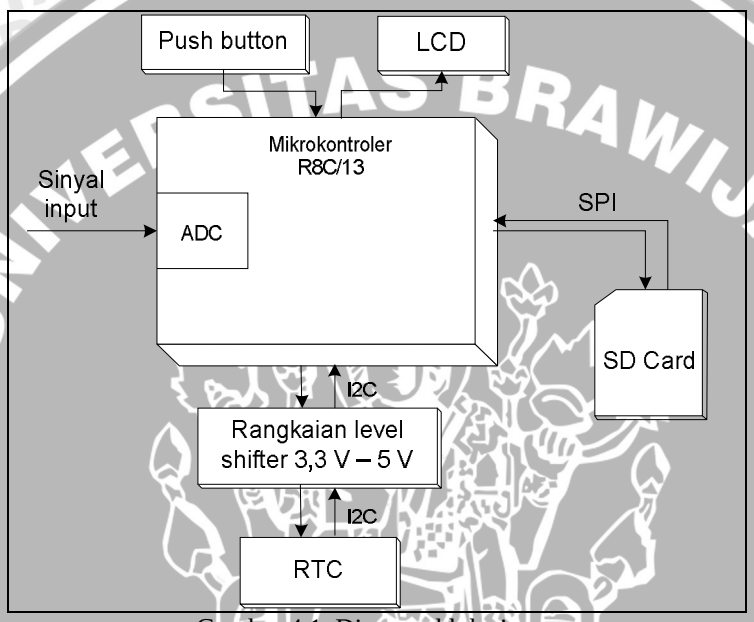
Sistem yang dirancang memiliki spesifikasi sebagai berikut:

- Mikrokontroler yang digunakan adalah R8C/13 yang telah memiliki *embedded* ADC.
- Basis waktu disediakan oleh IC RTC DS1307 yang memberikan informasi detik, menit hingga tahun.
- Data masukan yang dapat direkam adalah data sinyal analog DC dengan *range* tegangan antara 0 V sampai tegangan referensi ADC sebesar 3,3 V. Selain itu, data sinyal analog dapat terekam secara efektif hanya jika sinyal analog tersebut tingkat laju perubahan amplitudo maksimal 0,0275 V/detik.
- Jumlah Kanal ADC yang dapat digunakan dalam perekaman adalah dua kanal.
- Resolusi ADC yang digunakan adalah 10 bit.
- Menggunakan 4 buah *push button* sebagai pemilih selang waktu kecepatan sampling. Pilihan kecepatan sampling adalah 2 menit, 5 menit, 10 menit dan 30 menit.

- Data direkam dan disimpan dalam *SD card* yang berkapasitas 1 Gb dengan filesystem FAT16 dan format file teks (*.TXT). Satu file teks berisi rekaman historis data dalam satu hari penuh.
- LCD dot matrix 2×16 digunakan sebagai indikator dan unit penampil.

4.2 Diagram Blok Sistem

Berdasarkan spesifikasi yang telah ditentukan, maka sistem dapat digambarkan secara garis besar dalam sebuah diagram blok seperti yang ditunjukkan dalam Gambar 4.1.



Gambar 4.1. Diagram blok sistem

Alat terdiri atas mikrokontroler R8C/13, RTC DS1307, rangkaian *push button*, modul LCD dan *SD card*. Dalam sistem ini, Mikrokontroler R8C/13 dan *SD card* dicatu oleh sumber tegangan 3,3 V, sedangkan IC RTC DS1307 dicatu oleh sumber tegangan 5 V. Dengan level tegangan yang berbeda antara Mikrokontroler dan IC RTC DS1307, maka diperlukan rangkaian tambahan yang berfungsi untuk mengubah level tegangan 3,3 V menjadi 5 V dan juga sebaliknya. Rangkaian *push button* digunakan untuk memberikan input pilihan seberapa sering sampling data dilakukan.

IC RTC berfungsi untuk memberikan informasi data waktu detik, menit hingga tahun. Mikrokontroler R8C/13 memproses informasi waktu tersebut sehingga dalam periode tertentu akan dilakukan sampling data masukan. Masukan analog akan diolah oleh ADC yang tersedia di dalam Mikrokontroler R8C/13. Data yang telah berbentuk data digital akan disimpan dalam *SD card*. Modul LCD berfungsi sebagai unit penampil informasi waktu dan data yang sedang terekam.

4.3 Perancangan Perangkat Keras

Perancangan perangkat keras dalam pembuatan sistem ini terdiri atas perancangan rangkaian push button, rangkaian IC RTC (*Real Time Clock*), rangkaian *bidirectional level shifter*, LCD dan antarmuka SPI *SD card*. Sebagaimana telah disebutkan, unit pengolah utama yang digunakan adalah mikrokontroler R8C/13. Mikrokontroler yang digunakan tersebut telah berupa modul evaluasi HRS8000. Dengan penggunaan modul tersebut, maka perancangan dapat difokuskan pada rangkaian elektronik di luar *minimum system* mikrokontroler. Modul tersebut menyediakan pin-pin input output sejumlah 20 pin atau juga dapat berjumlah 22 pin jika kristal eksternal tidak dipergunakan. Komunikasi (*download firmware* dan *debugging*) R8C/13 dengan PC dilakukan melalui jalur serial RS232. Mikrokontroler R8C/13 dalam sistem ini dioperasikan dengan catu daya 3,3 volt dan menggunakan *clock* eksternal 20 MHz. Selain itu, tegangan referensi ADC juga diberi tegangan sebesar 3,3 volt.

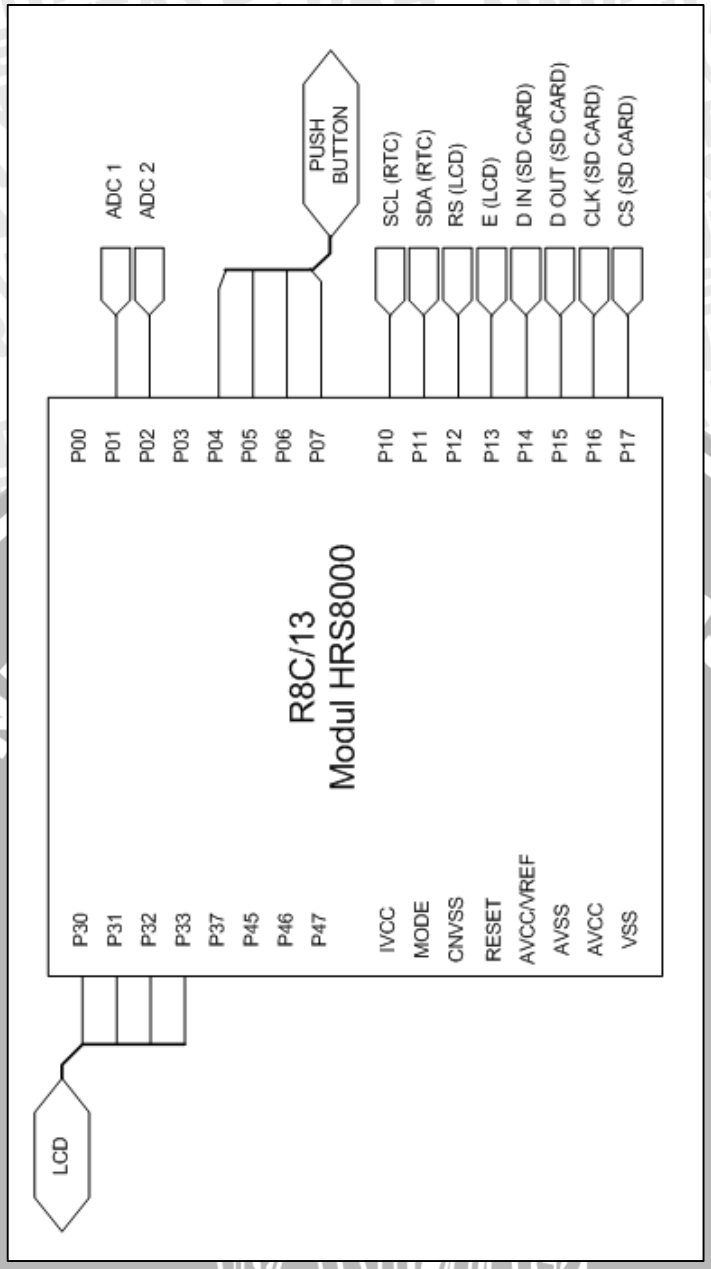
Pin-pin yang digunakan di dalam sistem ini beserta fungsinya adalah sebagai berikut:

A. PORT 0
- Pin P0_1 digunakan sebagai jalur ADC 1 (kanal 6)
- Pin P0_2 digunakan sebagai jalur ADC 2 (kanal 5)
- Pin P0_4 – P0_7 digunakan sebagai jalur data empat buah *push button*

B. PORT 1
- Pin P1_0 digunakan sebagai jalur SCL dari IC RTC
- Pin P1_1 digunakan sebagai jalur SDA dari IC RTC
- Pin P1_2 digunakan sebagai jalur RS dari LCD
- Pin P1_3 digunakan sebagai jalur E dari LCD
- Pin P1_4 digunakan sebagai jalur Data In (MISO)
- Pin P1_5 digunakan sebagai jalur Data Out (MOSI)
- Pin P1_6 digunakan sebagai jalur *clock* (SCK)
- Pin P1_7 digunakan sebagai jalur *chip select* (CS)

C. PORT 3
- Pin P3_0 – P3_3 digunakan sebagai jalur data LCD

Gambar 4.2 menunjukkan penggunaan pin-pin mikrokontroler di dalam sistem.

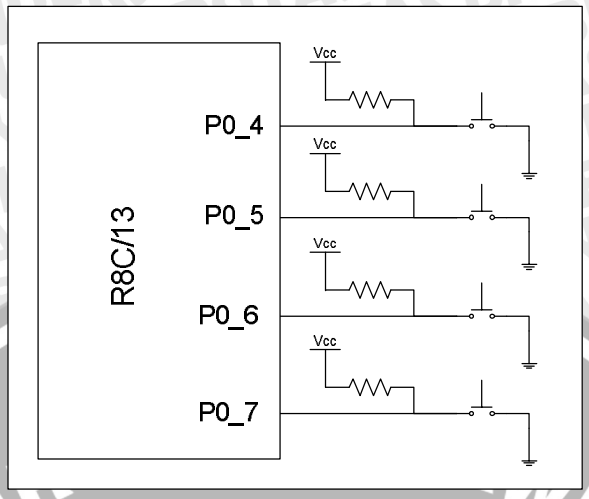


Gambar 4.2. Penggunaan pin-pin mikrokontroler di dalam sistem

4.3.1 Perancangan Rangkaian *Push Button*

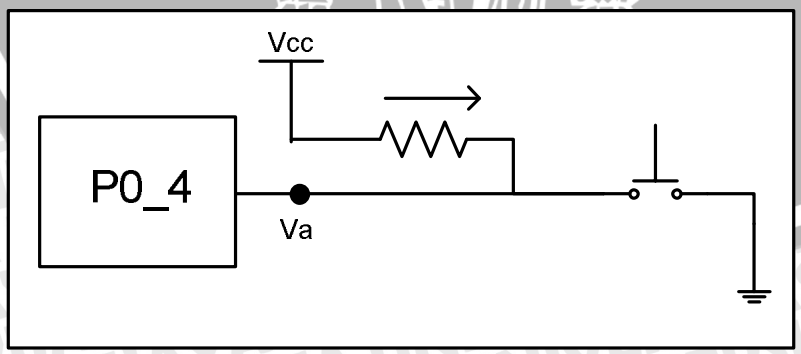
Push button di dalam sistem berfungsi sebagai tombol antarmuka yang digunakan untuk memilih kecepatan sampling ADC. Berdasarkan spesifikasi sistem yang telah dijelaskan sebelumnya, sistem memberikan empat macam pilihan kecepatan sampling, yaitu tiap 2 menit, 5 menit, 10 menit dan tiap 30 menit. Dengan begitu, maka *push button* yang digunakan berjumlah empat buah yang mewakili empat macam pilihan sampling. Tiap *push button* dihubungkan dengan pin mikrokontroler R8C/13 yang sebelumnya telah diatur sehingga berfungsi sebagai pin masukan. Di dalam

perancangan ini *push button* dihubungkan dengan pin P0_4 sampai P0_7. Gambar 4.3 menunjukkan rangkaian *push button*.



Gambar 4.3 Rangkaian *push button*

Pada saat *push button* ditekan, maka pin mikrokontroler akan terhubung langsung dengan ground sehingga terbaca sebagai logika 0. Namun pada kondisi *push button* tidak tertekan, akan terjadi logika ambang. Untuk memastikan logika pada saat kondisi tidak tertekan, maka dipasang resistor pull up. Pemilihan nilai resistor pull up didasarkan pada nilai arus maksimum yang dapat masuk mikrokontroler pada saat logika high (I_{IHmaks}) dan nilai tegangan minimum yang dikenali mikrokontroler sebagai logika 1 (V_{IHmin}). Berdasarkan *datasheet* diketahui bahwa I_{IHmaks} bernilai sebesar 4µA dan V_{IHmin} bernilai sebesar 0,8Vcc. Karena Vcc mikrokontroler yang digunakan sebesar 3,3 V, maka V_{IHmin} bernilai sebesar 2,64 V. Dalam analisis berikut dimisalkan resistor pull up yang digunakan bernilai 10kΩ. Gambar 4.4 menunjukkan analisis rangkaian resistor pull-up yang dipergunakan *push button*.



Gambar 4.4 Analisis rangkaian resistor pull up *push button*

Pada saat *push button* tidak ditekan, maka:

$V_a = Vcc - V_{IR}$ .....(1)

$V_a = 3,3 – (4µA)(10kΩ)$

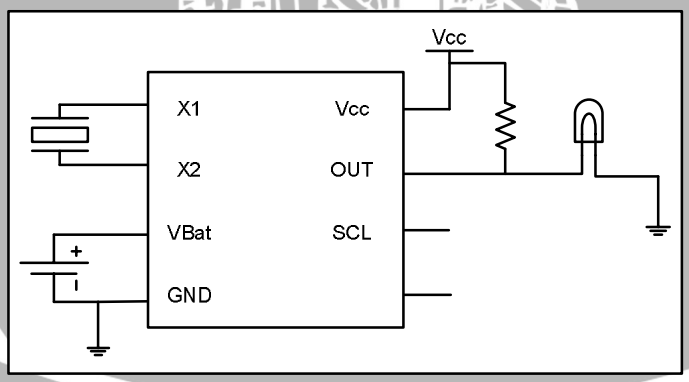
V_a= 3,3 – 0.004

V_a= 3,296 V

Dengan nilai resistor 10 kΩ didapatkan tegangan yang terbaca mikrokontroler sebesar 3,296 V pada saat logika *high*. Nilai ini masih lebih besar daripada tegangan minimum yang dianggap logika *high* oleh mikrokontroler yaitu 2,64 V. Dengan begitu, resistor sebesar 10 kΩ dapat digunakan.

4.3.2 Perancangan Rangkaian RTC DS1307

Sebagaimana telah dijelaskan dalam diagram blok sistem, penyedia informasi waktu dilakukan oleh IC RTC tipe DS1307. IC ini memiliki register-register yang berisi informasi waktu mulai dari detik, menit, jam, tanggal, hari ke-, bulan dan tahun. Selain itu, DS1307 dilengkapi register tambahan yang berfungsi untuk memberikan pulsa output dengan frekuensi tertentu. Secara fisik, DS1307 memiliki 8 kaki. Catu utama yang diberikan kepada IC ini sebesar 5 V. IC ini memerlukan Xtal eksternal sebesar 32,768 kHz. DS1307 memiliki pin yang dapat dihubungkan dengan baterai 3,3 agar sistem pewaktuan dapat terus berjalan meski catu utama dimatikan. Jenis komunikasi yang digunakan adalah I2C yang berarti cukup memerlukan dua jalur yang dihubungkan dengan mikrokontroler, yakni pin SDA yang berisi data dan pin SCL yang berfungsi sebagai clock. Pin SCL dihubungkan dengan pin P1_0, sedangkan pin SDA dihubungkan dengan pin P1_1 mikrokontroler. Gambar 4.5 menunjukkan rangkaian DS1307.



Gambar 4.5 Rangkaian RTC DS1307

Sebagai indikator bahwa DS1307 berjalan normal, dipasang LED pada kaki output yang sebelumnya telah diatur agar memberikan pulsa keluaran dengan frekuensi 1 Hz. Karena pin output bersifat *open drain*, maka diperlukan resistor pull up untuk memberikan logika *high* pada LED. Pada saat logika *high*, pin output akan bersifat ambang. Dengan begitu, maka arus akan mengalir melalui resistor pull up kemudian

melewati LED dan akhirnya menuju ground. Dengan merancang arus yang melewati LED adalah sebesar 15 mA, maka resistor pull up minimum yang dibutuhkan adalah sebesar:

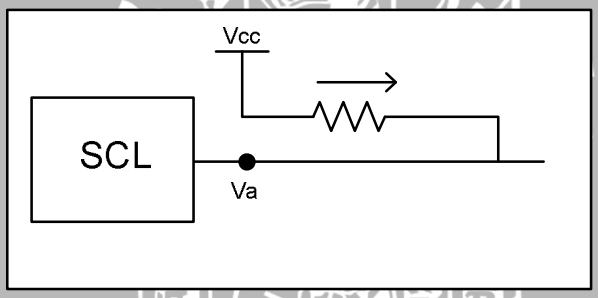
$$R = \frac{V}{I} = \frac{5}{15 \times 10^{-3}} = 333{,}333\Omega$$

Dalam perancangan digunakan resistor sebesar 1 kΩ. Dengan memberikan nilai tersebut, maka nilai arus didapatkan sebesar

$$I = \frac{V}{R} = \frac{5}{1 \times 10^{3}} = 5\ mA$$

Arus sebesar ini masih dapat menyalakan LED, sehingga resistor 1 kΩ dapat digunakan.

Kedua pin yang digunakan sebagai jalur komunikasi, yaitu SCL dan SDA harus dihubungkan dengan resistor pull up untuk memastikan kondisi logika. Hal ini disebabkan karena kedua pin tersebut juga bersifat *open drain*. Gambar 4.6 menunjukkan analisis resistor pull up pada pin SCL.



Gambar 4.6 Analisis resistor pull up output RTC

Berdasarkan *datasheet* diketahui bahwa I_{IHmaks} bernilai sebesar 1µA dan V_{IHmin} bernilai sebesar 2,2 V. Vcc yang digunakan RTC sebesar 5 V. Dalam analisis berikut dimisalkan resistor pull up yang digunakan bernilai 4,7 kΩ karena umum digunakan dalam komunikasi I2C.

$V_a = Vcc - V_{IR}$

$V_a = 5 – (1\mu A)(4{,}7k\Omega)$

$V_a = 5 – 0.0047$

$V_a = 4{,}9953$ V

Dengan nilai resistor 4,7 kΩ didapatkan tegangan yang terbaca RTC sebesar 4,9953 V pada saat logika *high*. Nilai ini masih lebih besar daripada tegangan minimum yang dianggap logika *high* oleh RTC yaitu 2,2 V. Sedangkan untuk sisi mikrokontoler:

$V_a = Vcc - V_{IR}$

$V_a = 3{,}3 – (4\mu A)(4{,}7\ k\Omega)$

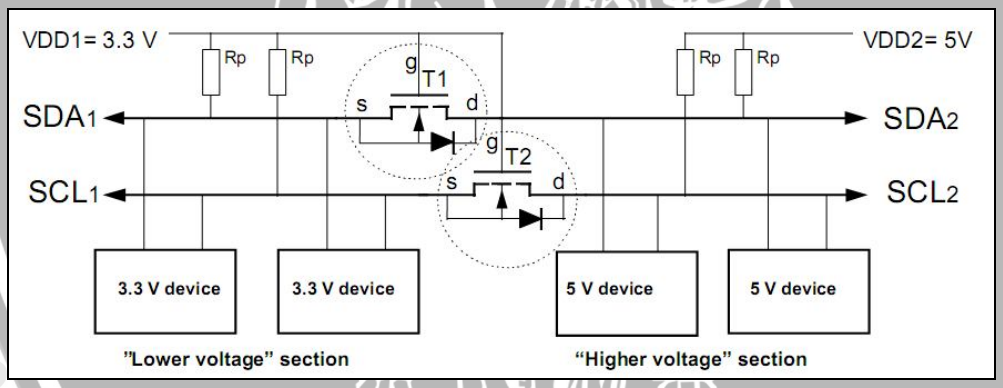
V_a= 3,3 – 0.0188

V_a= 3,2812 V

Dengan nilai resistor 4,7 kΩ didapatkan tegangan yang terbaca oleh mikrokontroler sebesar 3,2812 V pada saat logika *high*. Nilai ini masih lebih besar daripada tegangan minimum yang dianggap logika *high* oleh mikrokontroler yaitu 0,8Vcc atau 2,64 V. Dengan begitu, resistor sebesar 4,7 kΩ dapat digunakan.

4.3.3 Rangkaian *bidirectional level shifter*

Dengan catu daya mikrokontroler sebesar 3,3 V dan catu daya RTC sebesar 5 V, maka diperlukan rangkaian tambahan yang dapat mengkonversi tegangan 3,3 V menjadi 5 V atau sebaliknya agar komunikasi I2C antara mikrokontroler dan RTC berjalan normal. Philips semiconductor telah memberikan *application note* tentang *bidirectional level shifter* yang memang ditujukan khusus untuk mengatasi hal tersebut. Rangkaian tersebut memanfaatkan MOSFET yang dapat bekerja sebagai *switch*. MOSFET yang digunakan dalam perancangan ini adalah jenis IRF540 yang memiliki V_{GS} *tresshold* minimal sebesar 2 V untuk membuat drain dan source tersambung. Gambar 4.7 menunjukkan rangkaian *bidirectional level shifter*.



Gambar 4.7 Rangkaian *bidirectional level shifter*

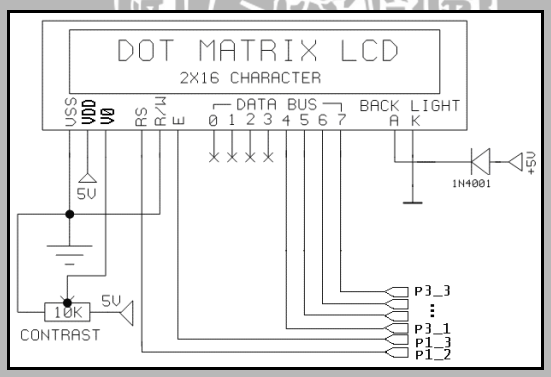
Analisis rangkaian tersebut adalah sebagai berikut:

- Keadaan 1. Tidak ada devais yang memberikan logika low. Maka, bus pada devais tegangan yang lebih rendah akan di pull-up oleh resistor Rp menjadi tegangan sekitar 3,3 V. Tegangan gate dan source pada MOSFET berada pada tegangan 3,3 V. Karena V_{GS} *tresshold* minimal pada MOSFET adalah 2 V, maka V_{GS} belum terpicu dan MOSFET belum aktif. Akibatnya, bus pada devais tegangan lebih tinggi akan di pull up oleh resistor Rp menjadi sekitar 5 V. dengan begitu, kedua bus akan berada pada logika *high* dengan level tegangan berbeda.

- Keadaan 2. Devais 3,3 V memberikan logika *low*. Kaki source MOSFET akan juga berlogika *low*, sedangkan gate tetap terpicu 3,3 V. Karena tegangan V_{GS} telah melebihi tegangan *tresshold*, maka MOSFET aktif dan drain-source tersambung. Akibatnya bagian drain juga berlogika *low* yang selanjutnya terbaca sebagai logika *low* oleh devais 5 V. jadi, kedua bus berada dalam kondisi tegangan yang sama saat berlogika *low*.
- Kondisi 3. Devais 5 V memberikan logika *low*. Melalui dioda subtrat drain pada MOSFET, bagian tegangan yang lebih rendah awalnya akan di pull down sampai V_{GS} melewati *tresshold* dan MOSFET menjadi aktif. Kemudian bus pada tegangan yang lebih rendah akan benar-benar di pull down menjadi logika low oleh devais 5 V melalui MOSFET yang telah terkonduksi. Jadi kedua devais telah berada pada logika *low* yang sama.

4.3.4 Rangkaian *Liquid Crystal Display* (LCD)

LCD 2×16 umumnya dapat dioperasikan dalam mode 8 bit atau 4 bit. Untuk menghemat penggunaan pin, perancangan ini mempergunakan mode 4 bit. Pin V0 dhubungkan dengan potensiometer yang berfungsi untuk mengubah tegangan kontras layar. Jalur data D7 – D4 LCD secara urut dihubungkan dengan port P3_3 – P3_0. Sedangkan pin RS dan E masing-masing dihubungkan dengan P1_2 dan P1_3. Gambar 4.8 menunjukkan rangkaian LCD.

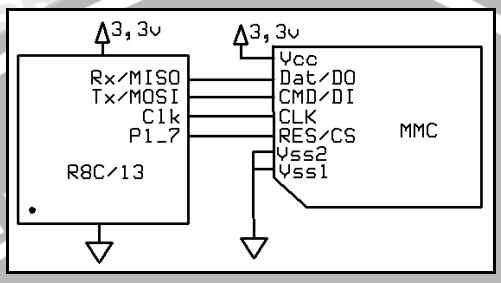


Gambar 4.8. Rangkaian LCD

4.3.5 Antarmuka SPI *SD card* dengan R8C/13

Mikrokontroler R8C/13 tidak memiliki pin antarmuka khusus SPI yang diperlukan untuk berkomunikasi dengan *SD card*. Namun, mikrokontroler R8C/13 memiliki antarmuka serial USART yang dapat diimplementasikan sebagai antarmuka SPI. Di dalam komunikasi SPI, terdapat dua bagian penting yakni *master* dan *slave*. *Master* bertindak sebagai peng-inisiasi awal komunikasi dengan *slave*. Dalam hal ini,

mikrokontroler R8C/13 bertugas sebagai *master* yang harus meyediakan sinyal *clock*. Pin Rx pada mikrokontroler R8C/13 diimplementasikan sebagai jalur MISO (*Master In Slave Out*) dan pin Tx diimplementasikan sebagai jalur MOSI (*Master Out Slave In*). *SD card* bekerja dengan catu daya 3,3 volt. Sementara itu, mikrokontroler R8C/13 dapat bekerja pada *range* 2,7 – 5 volt. Oleh karena itu, keduanya dioperasikan dengan catu daya yang sama yakni 3,3 volt. implementasi *hardware* komunikasi USART sebagai komunikasi SPI ditunjukkan dalam Gambar 4.9.



Gambar 4.9 Implementasi *hardware* antarmuka USART sebagai SPI

4.4 Perancangan Perangkat Lunak

Di dalam peracangan sistem ini, terdapat dua bagian perancangan *software*. Pertama, perancangan perangkat lunak untuk *firmware* mikrokontroler R8C/13 sehingga dapat berfungsi menjadi unit pemroses dan pengendali utama seluruh *hardware* sistem. Kedua, perancangan perangkat lunak untuk komputer yang berfungsi sebagai *software* pembantu untuk menampilkan data historik yang telah terekam di dalam *SD card*.

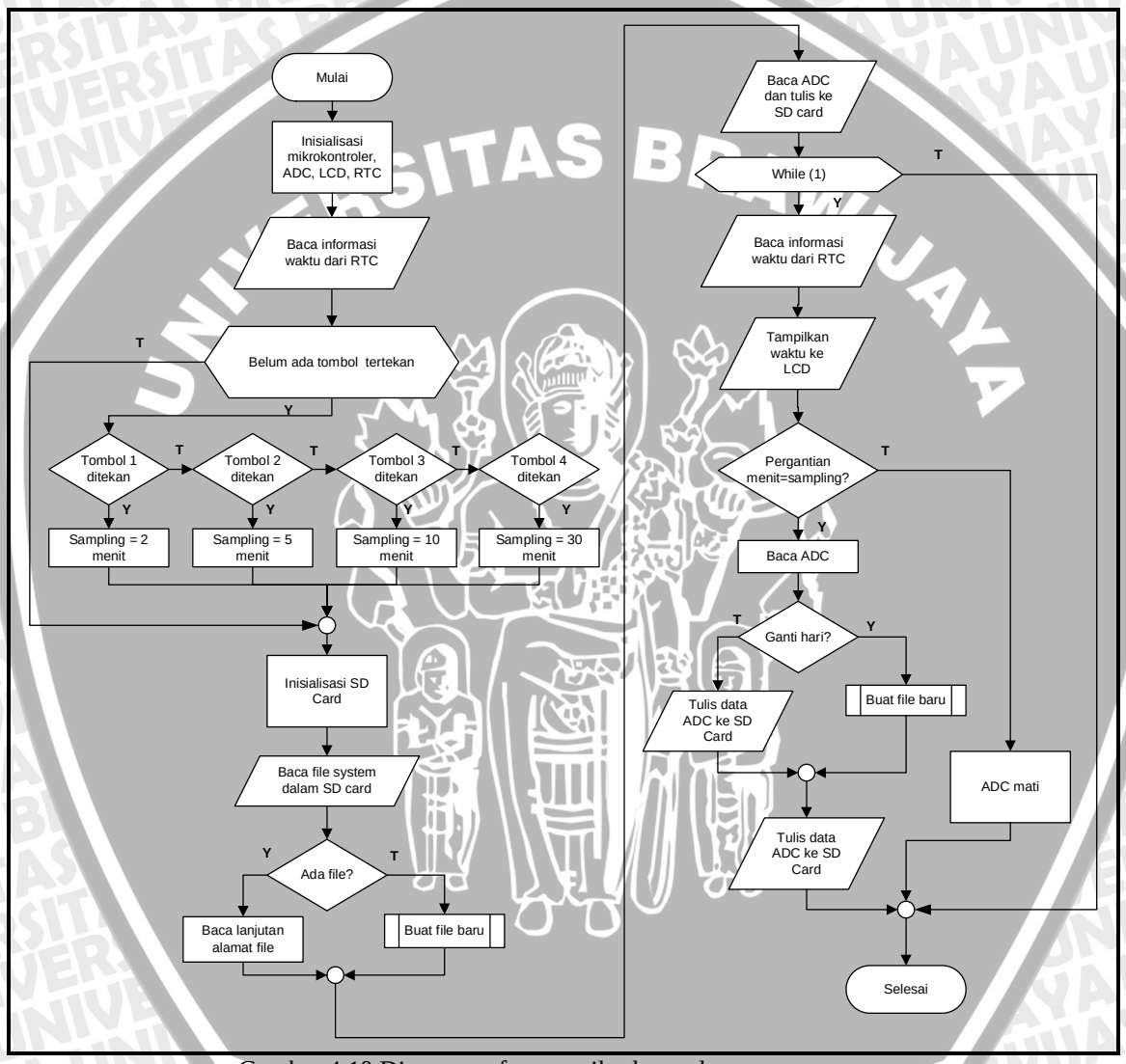
Perancangan *firmware* mikrokontroler meliputi perancangan perangkat lunak yang berhubungan dengan pengaturan *hardware* sistem yakni inisiasi mikrokontroler R8C/13 itu sendiri, pengaturan komunikasi dengan *SD card*, pengendalian LCD dan *push button*, pembacaan sinyal yang masuk ADC, serta komunikasi dengan IC *Real Time Clock*. Selain menangani pengaturan *hardware*, *firmware* mikrokontroler juga dirancang untuk menangani lapisan *filesystem*. Dalam lapisan ini, mikrokontroler bertugas untuk menangani proses pembuatan *file* yang dapat terbaca di dalam sebuah sistem operasi, yakni antara lain analisis struktur *filesystem*, pencarian *cluster* kosong, serta proses pembuatan dan penamaan file. Pembuatan perangkat lunak sekaligus *downloader* untuk mikrokontroler dilakukan dengan bantuan software HEW yang berbasis bahasa pemrograman C.

Perancangan perangkat lunak untuk komputer dilakukan dengan bantuan *software* Delphi yang berbasis bahasa Pascal. Program yang dirancang tersebut

berfungsi untuk menampilkan data historik yang terekam di dalam *SD card* dalam bentuk tabel sekaligus grafik yang bertujuan untuk mempermudah analisis data.

4.4.1 Perancangan Program Utama

Program utama merupakan program yang pertama kali dijalankan oleh mikrokontroler yang kemudian dapat memanggil atau menjalankan subprogram yang lain. Perancangan program utama tersebut ditunjukkan oleh *flowchart* dalam Gambar 4.10.



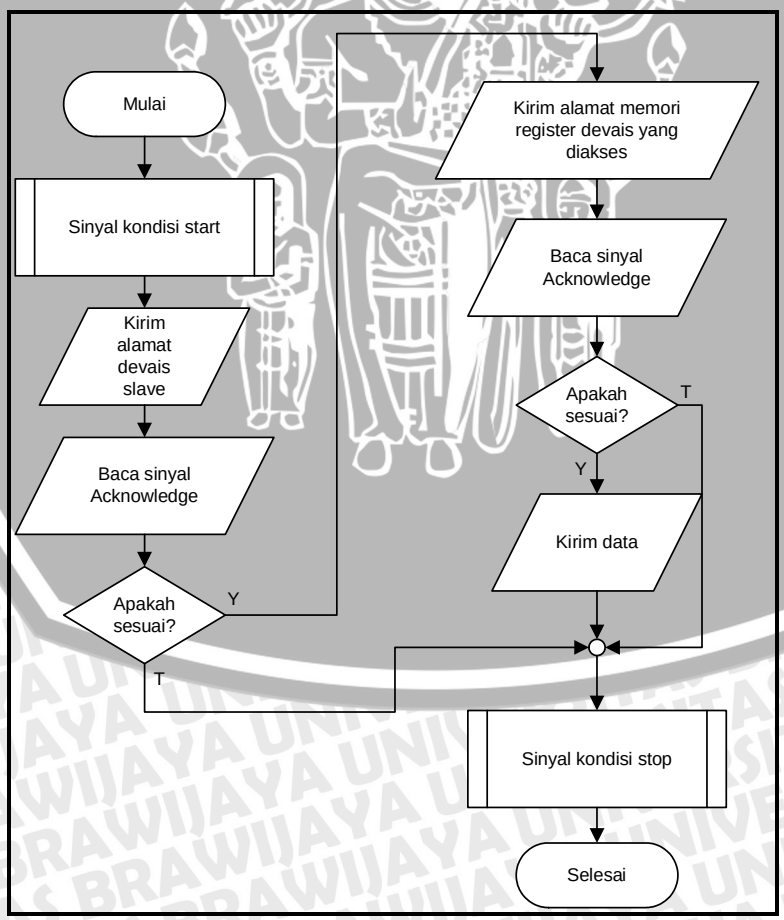
Gambar 4.10 Diagram software mikrokontroler secara umum

Pada saat sistem dihidupkan, mikrokontroler melakukan beberapa inisialisasi *hardware* yang meliputi inisialisasi mikrokontroler, RTC dan LCD. Selanjutnya, mikrokontroler membaca input dari *push button*. Input tersebut digunakan sebagai pemilihan seberapa sering sampling data ingin dilakukan. Setelah itu, mikrokontroler melakukan inisialisasi *SD card*. Proses selanjutnya, mikrokontroler mengambil sampling awal sinyal ADC dan kemudian membuat *file* di dalam *SD card*. Kemudian

mikrokontroler membaca data waktu dari RTC. Jika data waktu belum menunjukkan selang waktu yang sesuai dengan pilihan pengguna, maka mikrokontroler terus membaca data waktu dari RTC. Jika data waktu telah menunjukkan selang yang sesuai, maka akan dicek apakah tanggal sudah berubah. Jika tanggal sudah berubah, maka akan dibuat *file* teks baru. Namun jika tanggal belum berubah ataupun file baru telah dibuat, maka mikrokontroler akan membaca data register ADC yang telah melakukan konversi sebelumnya. Data hasil konversi ADC tersebut akan ditulis ke dalam *SD card* dalam format *file* teks sehingga dapat dibaca menggunakan PC. Data waktu dan data hasil konversi juga ditampilkan melalui LCD. Program akan berjalan *looping* seterusnya selama sistem aktif.

4.4.2 Perangkat Lunak Unit RTC DS1307

Sebagaimana yang telah dijelaskan sebelumnya, unit pemroses dan penyedia informasi waktu ditangani oleh IC RTC DS1307. Untuk dapat mengakses perangkat RTC ini, mikrokontroler mempergunakan protokol komunikasi I2C yang dikembangkan oleh phillips. Proses transfer data dapat digambarkan dalam bentuk *flowchart* yang ditunjukkan dalam Gambar 4.11.



Gambar 4.11 Flowchart software protokol transfer data I2C

Sinyal kondisi start dilakukan dengan cara memberikan output *high* pada pin clock SCL dan sekaligus memberikan logika tepi turun pada pin data SDA. Sedangkan kondisi stop dilakukan dengan cara memberikan output *high* pada pin clock SCL dan sekaligus memberikan logika tepi naik pada pin data SDA. Setiap data yang dikrim lewat pin SDA adalah berukuran 1 byte selama 8 clock yang selanjutnya dibalas oleh *acknowledge* selama 1 clock. Saat inisiasi pertama akan dilakukan pencocokan alamat *slave*. Alamat untuk DS1307 adalah 7 bit biner 1101000 yang kemudian ditambahi 1 bit akhir berupa mode tulis atau baca. Untuk proses tulis, maka bit kedelapan adalah 0. Sedangkan untuk proses baca, maka bit kedelapan adalah 1. Jika alamat tersebut cocok, maka selanjutnya adalah pengiriman alamat register yang akan diakses. Tabel 4.1 menunjukkan register waktu yang dapat diakses dan dimodifikasi.

Tabel 4.1. Alamat dan isi register DS1307

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	X	10 MINUTES			MINUTES				00–59
	X	12/24	10 HR A/P	10 HR	HOURS				01–12 00–23
	X	X	X	X	X	DAY			1–7
	X	X	10 DATE		DATE				01–28/29 01–30 01–31
	X	X	X	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

Sumber: Dallas Semiconductor, 2000: 4

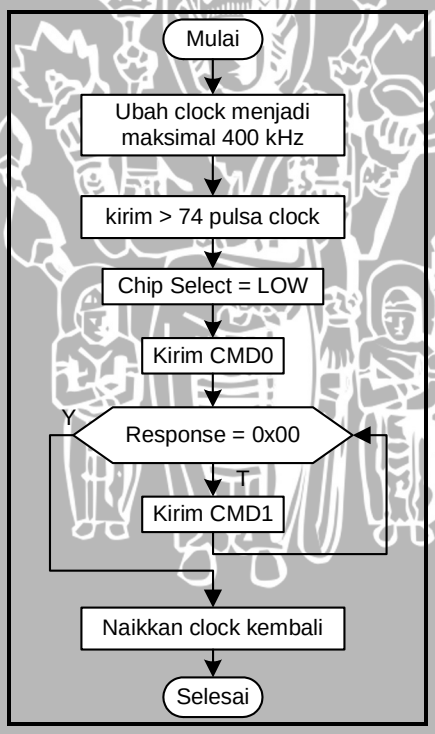
4.4.3 Perangkat Lunak Unit *SD card*

Pada dasarnya *SD card* adalah sama seperti jenis unit memori lain seperti EEPROM. *SD card* juga tersusun atas larik-larik memori yang telah diberi alamat tertentu. Untuk dapat mengakses *SD card*, mikrokontroler harus dapat mengerti protokol komunikasi yang digunakan oleh *SD card*. *SD card* menyediakan dua macam mode komunikasi yakni mode SD dan mode SPI. Mode SPI adalah mode yang dikenal secara luas di dalam dunia pemrograman dan antarmuka menggunakan mikrokontroler. Oleh karena itu, perancangan sistem ini memanfaatkan mode SPI untuk melakukan komunikasi dengan *SD card*.

4.4.3.1 Inisialisasi mode SPI

Untuk mempergunakan protokol komunikasi mode SPI, perlu dilakukan beberapa prosedur.

Ketika pertama kali dihidupkan, tunggu beberapa milisekon hingga *SD card* memasuki mode operasi aslinya (protokol SD bus). Setelah itu set DI dan CS dengan logika *high* dan berikan sedikitnya 74 pulsa *clock* pada pin SCLK. Kirimkan perintah CMD0 dengan sinyal CS *low* untuk me-reset *SD card*. Pada saat CMD0 diterima, *SD card* memasuki kondisi *idle* dan respons R1 adalah 0x01 (bit *In-Idle-State* bernilai 1). Dalam keadaan *idle, SD card* hanya dapat menerima perintah CMD0, CMD1, dan CMD58. Jika pada saat *idle SD card* menerima perintah CMD1, maka inisialisasi akan dimulai. Untuk mengetahui akhir inisialisasi, *master* harus melakukan cek terhadap respons. Inisialisasi sudah selesai jika respons R1 kembali bernilai 0x00 (bit *In-Idle-State* bernilai 0. Setelah inisialisasi, operasi baca/tulis dalam mode SPI dapat dilakukan. *Flowchart* inisialisasi mode SPI untuk *SD card* ditunjukkan dalam Gambar 4.12



Gambar 4.12 *Flowchart* Inisialisasi mode SPI untuk *SD card*

4.4.3.2 Pembuatan file dan penulisan data

Sebagaimana yang telah dijelaskan dalam bagian Tinjauan Pustaka untuk *filesystem* FAT, secara global *filesystem* FAT16 terdiri atas *reserved area*, FAT area dan data area. *Reserved area* hanya berisi beberapa informasi mengenai *SD card* yang

bersangkutan. Untuk membuat dan memodifikasi sebuah file, bagian yang perlu dicermati adalah FAT area dan data area.

FAT area adalah sebuah struktur di dalam *filesystem* yang menyatakan alokasi sebuah file. Jika sebuah file dialokasikan lebih dari satu *cluster* maka alamat *cluster* selanjutnya dapat ditemukan dalam area ini. Struktur FAT digunakan untuk mengetahui lokasi *cluster* selanjutnya dari sebuah file dan juga status *cluster* apakah sudah terpakai atau belum.

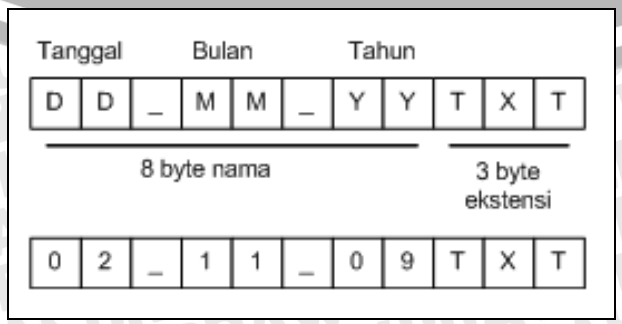
Data area tersusun atas *root directory* dan data area itu sendiri. *Root directory* berisi penamaan sebuah file beserta segala atributnya dan juga alamat *cluster* yang digunakan di dalam data area.

Prosedur dalam perancangan pembuatan file dilakukan dalam beberapa tahap.

1. Mencari *cluster* kosong
2. Memesan tempat di *root directory* berupa penamaan file, ekstensi, atribut dan alamat yang digunakan
3. Menulis file di data area sesuai dengan alamat yang telah didefinisikan di *root directory*
4. Meng-*update* tabel FAT yang menunjukkan *cluster* tersebut telah dipergunakan

Untuk sebuah file, *root directory* mengalokasikan 32 byte data sebagai atribut file. Pengisian 32 byte data tersebut dapat mengacu pada Tabel 2.5 di dalam Bab Tinjauan pustaka. Byte yang perlu diperhatikan adalah byte penamaan file, penanggalan file, alamat file dan ukuran file.

Penamaan file dalam *root directory* menggunakan format *short file name* (format 8.3). Format 8.3 mensyaratkan nama file sebanyak 8 byte dan ekstensi 3 byte. Sebagaimana yang dicantumkan dalam Tabel 2.5 di dalam bab Tinjauan pustaka, penamaan file dengan format 8.3 memanfaatkan 11 byte awal dari total 32 byte yang disediakan. Gambar 4.13 menunjukkan format penamaan file yang dirancang untuk sistem ini.



Gambar 4.13 Format penamaan 8.3

Penamaan file dibuat berdasarkan waktu file dibuat. Di dalam Gambar 4.12 dimisalkan bahwa file dibuat pada tanggal 2 November 2009. Maka file akan muncul dengan nama 02_11_09.TXT di dalam *SD card*.

Penyedia informasi waktu kapan file dibuat dan dimodifikasi disusun oleh masing-masing 16 bit data. Format tanggal dan susunan bit-bitnya ditunjukkan dalam Tabel 4.2.

Tabel 4.2 Format bit-bit penanggalan

Bit ke-	15 - 9	8 - 5	4 - 0
Isi data	Tahun. Berisi nilai 0 – 127. 0 adalah tahun 1980.	Bulan. Berisi nilai 1 – 12.	Tanggal. Berisi nilai 1 – 31.

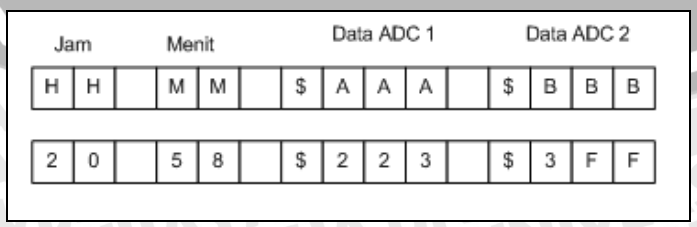
Sedangkan format waktu berupa detik, menit dan jam ditunjukkan dalam Tabel 4.3.

Tabel 4.3 Format bit-bit pewaktuan

Bit ke-	15 - 11	10 - 5	4 - 0
Isi data	Jam. Berisi nilai 0 – 23.	Menit. Berisi nilai 0 – 59.	Detik/2. Berisi nilai 0 – 29.

Alamat *cluster* yang digunakan sebuah file ditunjukkan oleh 16 bit data yang ada pada byte ke-27 dan 28 dari total 32 byte entri *root directory*. Sedangkan ukuran file dibentuk dari 4 byte akhir dari total 32 byte entri *root directory*. Khusus untuk byte penanggalan, pewaktuan, alamat dan ukuran file, susunan byte dibalik. Byte dengan nilai terkecil diletakkan didepan, baru kemudian byte dengan nilai terbesar.

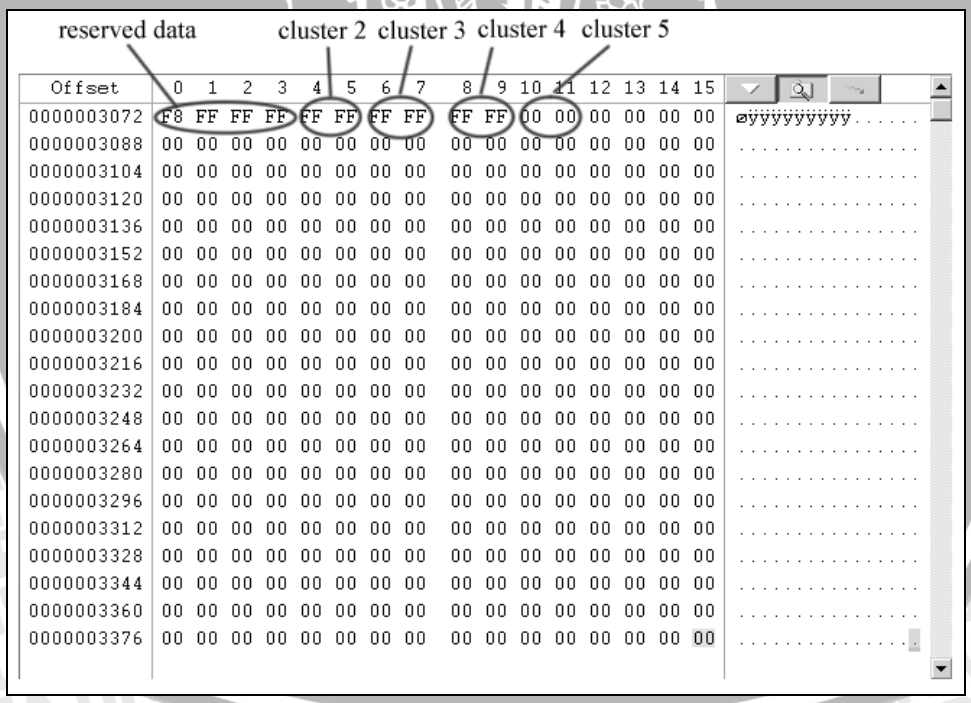
Data yang disimpan di dalam data area adalah data yang nantinya muncul di dalam file teks. Informasi yang penting untuk ditampilkan di dalam file teks meliputi informasi waktu berupa jam dan menit kapan data disampling, nilai sampling dari ADC kanal pertama dan nilai ADC dari kanal kedua dalam bentuk bilangan hexadesimal. Format data yang disimpan di dalam data area ditunjukkan di dalam Gambar 4.14.



Gambar 4.14 Format data di dalam data area

Di dalam Gambar 4.14 terlihat bahwa dalam setiap sampling, informasi data yang diambil meliputi jam, menit, data ADC 1 dan data ADC 2. Di dalam Gambar tersebut dimisalkan bahwa data disampling pada pukul 20.58 dengan nilai ADC 1 sebesar $223 atau 547 desimal dan nilai ADC 2 sebesar $3FF atau 1023 desimal. Oleh karena nilai yang ditunjukkan di dalam file teks berupa nilai ASCII, maka nilai data harus dikonversi telebih dahulu ke dalam nilai ASCII. Untuk data berupa angka, nilai ASCII dapat diperoleh dengan menjumlahkan angka tersebut dengan 0x30. Sedangkan data berupa huruf, nilai ASCII dapat diperoleh dengan menjumlahkan nilai angka dengan 0x37.

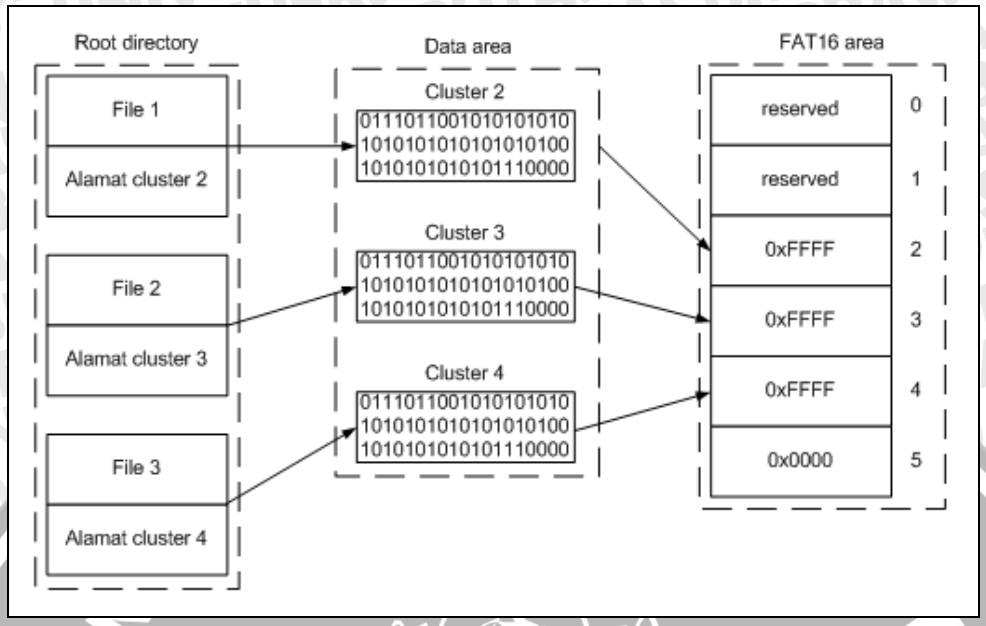
Tiap kali selesai membuat sebuah file yang ditempatkan pada alamat *cluster* tertentu, tabel FAT harus selalu di-*update*. Hal ini dilakukan untuk proses pencarian *cluster* kosong pada saat file pertama kali akan dibuat. Tabel FAT itu sendiri juga terdiri atas sector-sector data. Empat byte awal sebuah tabel FAT berisi *reserved data*. *Update cluster* data yang digunakan file dimulai pada byte ke 5 tabel FAT. Gambar 4.15 menunjukkan struktur tabel FAT.



Gambar 4.15 struktur tabel FAT

Dengan digunakannya *filesystem* FAT16, maka alamat *cluster* sebuah file akan ditunjukkan oleh 16 bit data. File yang dibuat di dalam sistem ini tidak melebihi ukuran 1 *cluster* atau 16 kb. Dengan begitu, maka 16 bit *cluster* penunjuk alamat file dalam tabel FAT langsung diisi dengan 0xFFFF yang berarti bahwa file hanya

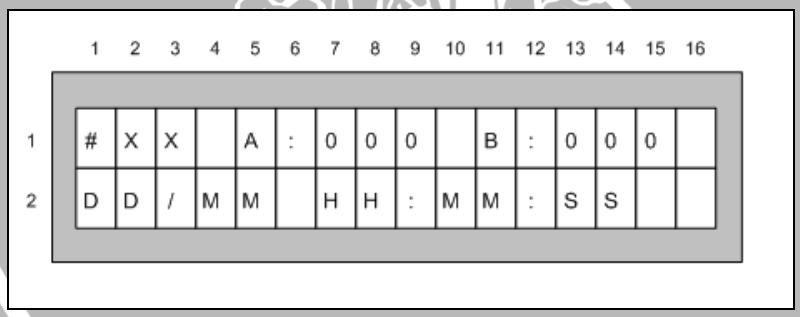
mempergunakan 1 *cluster*. Hubungan *root directory*, *cluster* data area dan tabel FAT untuk sebuah file dengan ukuran 1 *cluster* ditunjukkan dalam Gambar 4.16.



Gambar 4.16 Hubungan root directory, data area dan tabel FAT

.4.4.4 Perancangan Tampilan LCD

LCD dirancang untuk dapat menampilkan beberapa informasi antara lain nilai ADC yang terbaca oleh sistem, pewaktuan sistem dan kecepatan sampling yang sedang digunakan. Dengan penggunaan LCD 2 baris 16 kolom, maka perancangan tampilan LCD ditunjukkan dalam Gambar 4.17.



Gambar 4.17 Perancangan tampilan LCD

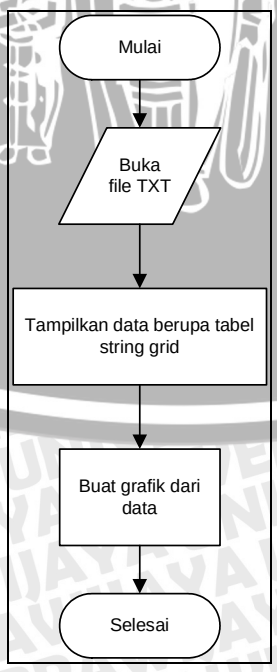
- *#XX* adalah nilai sampling yang dipilih pengguna dan sedang digunakan. Nilai yang ditunjukkan dapat berupa 02, 05, 10 dan 30.
- *A:000* adalah nilai ADC yang terbaca pada kanal pertama dalam bilangan hexadesimal. Bernilai antara 000 sampai 3FF.
- *B:000* adalah nilai ADC yang terbaca pada kanal kedua dalam bilangan hexadesimal. Bernilai antara 000 sampai 3FF.

- *DD* menunjukkan tanggal saat ini. Bernilai antara 01 sampai 31 dan tergantung jumlah hari dalam bulan tersebut.
- *MM* menunjukkan bulan saat ini. Bernilai antara 01 sampai 12.
- *HH* menunjukkan jam saat ini. Bernilai antara 00 sampai 23.
- *MM* menunjukkan menit saat ini. Bernilai antara 00 sampai 59.
- *SS* menunjukkan detik saat ini. Bernilai antara 00 sampai 59.

4.4.5 Perancangan *software* pada komputer

Data yang terekam di dalam *SD card* adalah berupa file teks (*.TXT). Di dalamnya terekam sejumlah angka yang menunjukkan waktu dan nilai ADC yang terekam sepanjang waktu sistem dinyalakan. Untuk mempermudah pembacaan kumpulan angka tersebut, akan dirancang pula *software* yang dapat membuat tabulasi data dan dapat dengan jelas menunjukkan karakteristik pola data. Dengan begitu, *software* yang dirancang berisi tabel dan grafik yang mengacu pada file hasil rekaman sistem.

Pembuatan *software* dilakukan dengan bahasa pemrograman Delphi. Di dalam Delphi, *software* tersusun atas beberapa form dan komponen. Form utama dirancang untuk memilih dan membuka file. Selain itu, form utama juga diisi komponen tabel untuk menampilkan tabulasi data. Form tambahan dibuat untuk dapat menampilkan komponen grafik dari data yang ditampilkan dalam tabel. Secara global, *flowchart* pemrograman *software* komputer ditunjukkan dalam Gambar 4.18.



Gambar 4.18 *Flowchart* pemrograman *Software* untuk komputer

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui tingkat kesesuaian hasil perancangan dan pembuatan alat dengan yang diharapkan. Pengujian dilakukan tiap blok perancangan untuk memudahkan analisis data. Pengujian yang dilakukan banyak memanfaatkan fasilitas *on-chip debugging* yang disediakan dalam HEW (*High-performance Embedded Workshop*), software pengembangan bawaan Renesas. Fasilitas *on-chip debugging* memungkinkan seseorang untuk memantau isi register mikrokontroler secara *real-time* ketika mikrokontroler sedang beroperasi. Untuk mempermudah pengecekan keberhasilan sistem, maka pengujian dilakukan perblok dan kemudian dilakukan pengujian secara keseluruhan. Pengujian tiap blok yang dilakukan adalah sebagai berikut:

a. Pengujian ADC
b. Pengujian *push button*
c. Pengujian Baca-Tulis *SD card*
d. Pengujian Modul LCD
e. Pengujian Rangkaian *Level shifter* 3,3 V-5 V
f. Pengujian RTC
g. Pengujian keseluruhan sistem

5.1 Pengujian ADC

5.1.1 Tujuan

Pengujian ini dilakukan untuk menganalisis apakah fasilitas *Analog to Digital Converter* yang disediakan mikrokontroler R8C/13 bekerja dengan baik

5.1.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000 yang dilengkapi simulator ADC berupa potensiometer sebagai pengatur tegangan masukan ADC
b. Software HEW

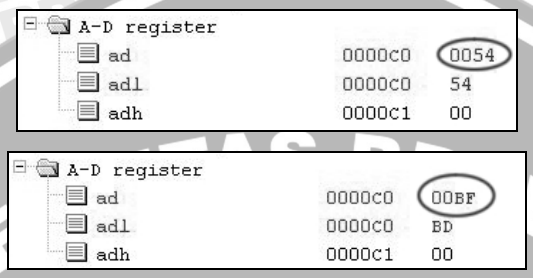
5.1.3 Prosedur Pengujian

a. Membuat program mikrokontroler Renesas R8C/13 untuk membaca masukan ADC dari kanal 6 dan 7 secara bergantian

b. Menghidupkan sistem, mengubah nilai tegangan ADC menggunakan potensiometer dan mengamati register ADC melalui program HEW

5.1.4 Hasil Pengujian dan Analisis

Dalam pengujian ini, mikrokontroler beserta fasilitas ADC telah dapat membaca masukan ADC dari tegangan yang diubah-ubah menggunakan potensiometer. Nilai ADC diamati melalui software HEW secara *realtime*. Tampilan pembacaan register ADC melalui *software* HEW ditunjukkan dalam Gambar 5.1.



Gambar 5.1 Nilai register ADC dengan tegangan masukan ADC yang berubah-ubah

Pengujian dilakukan beberapa kali untuk pembacaan nilai ADC dengan memberikan tegangan masukan yang berbeda. Dengan membandingkan penghitungan nilai ADC secara teori, didapatkan besar error untuk tiap pembacaan nilai ADC. Tabel 5.1 menunjukkan perhitungan error pembacaan ADC.

$$\text{Nilai teori} = \frac{\text{nilai desimal}}{\text{nilai tegangan referensi maksimum}} \times \text{Nilai desimal maksimum} \quad \text{.....(1)}$$

$$\text{Error} = \frac{\text{selisih nilai desimal dan teori}}{\text{nilai teori}} \times 100\ \% \quad \text{.....(2)}$$

Tabel 5.1 Perhitungan error pembacaan ADC

Volt	Nilai Hexa	Nilai Desimal	Teori	Error
0,29	56	86	86,24	0,28 %
0,49	91	145	145,72	0,49 %
1,1	147	327	327,12	0,04 %
1,78	212	530	529,34	0,12 %
2,26	2A3	675	672,09	0,43 %
3,44	3FF	1023	1023	0 %

5.2 Pengujian *push button*

5.2.1 Tujuan

Pengujian ini dilakukan untuk menganalisis apakah rangkaian *push button* dapat terbaca dengan baik oleh fasilitas *programmable input-output* yang disediakan mikrokontroler R8C/13.

5.2.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000

b. Software HEW

c. Rangkaian *push button*

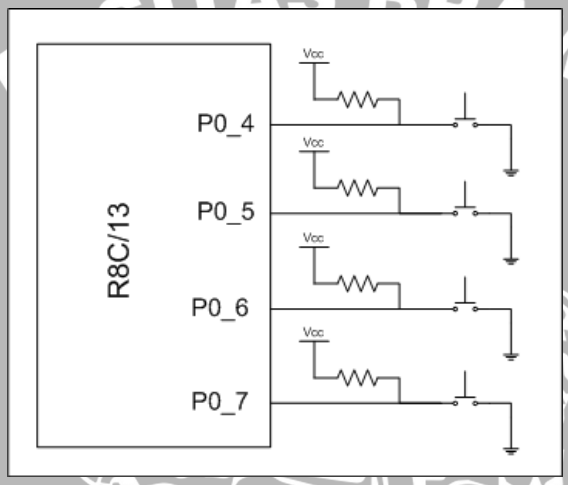
5.2.3 Prosedur Pengujian

a. Membuat program mikrokontroler Renesas R8C/13 untuk membaca masukan melalui port 0 (P_0)

b. Menghidupkan sistem, menekan push button secara bergantian dan mengamati register P_0 melalui program HEW

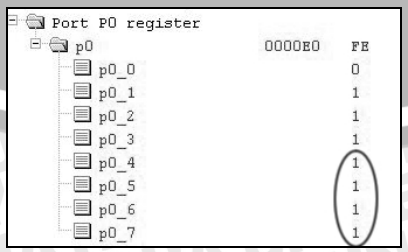
5.2.4 Hasil Pengujian dan Analisis

Rangkaian pengujian pembacaan push button melalui mikrokontroler R8C/13 ditunjukkan dalam Gambar 5.2.

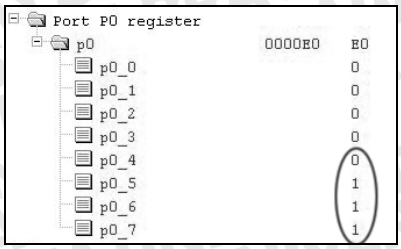


Gambar 5.2 Rangkaian pengujian pembacaan push button melalui mikrokontroler R8C/13

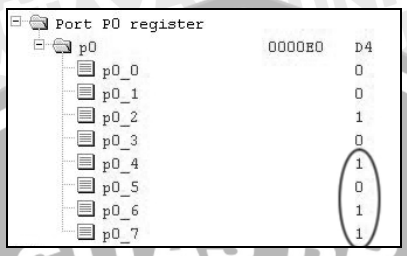
Dalam pengujian ini, mikrokontroler telah dapat membaca masukan *push button* dengan baik melalui port 0. Nilai register port 0 diamati melalui software HEW secara *realtime*. *Push button* dirancang dalam bentuk *normally closed* dan berlogika tinggi. Jadi, saat *push button* ditekan, akan mengirimkan logika 0 kepada mikrokontroler. Tampilan pembacaan register port 0 melalui software HEW ditunjukkan dalam Gambar 5.3 sampai Gambar 5.7.



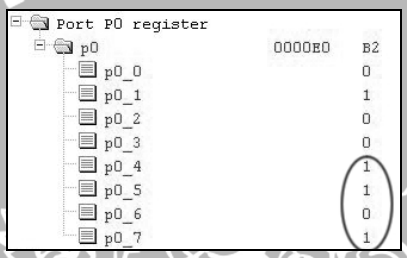
Gambar 5.3 Register p0 saat tidak ada *push button* yang tertekan



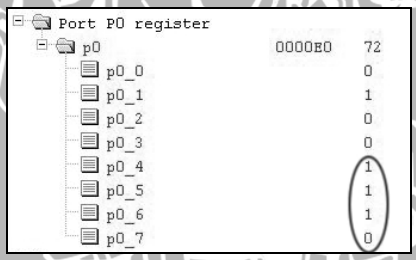
Gambar 5.4 Register p0 saat *push button* pada p0_4 tertekan



Gambar 5.5 Register p0 saat *push button* pada p0_5 tertekan



Gambar 5.6 Register p0 saat *push button* pada p0_6 tertekan



Gambar 5.7 Register p0 saat *push button* pada p0_7 tertekan

5.3 Pengujian Baca-Tulis *SD card*

5.3.1 Pengujian Tulis *SD card*

5.3.1.1 Tujuan

Pengujian ini dilakukan untuk menganalisis apakah mikrokontroler dapat menulis ke dalam *SD card* dengan baik.

5.3.1.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000

b. *SD card*

c. *Software* HEW

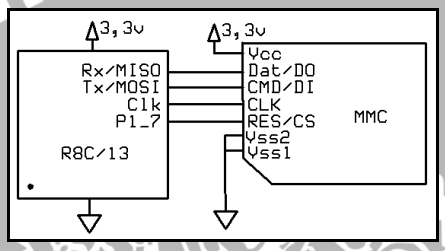
d. *Software* WinHex

5.3.1.3 Prosedur Pengujian

a. Membuat program mikrokontroler Renesas R8C/13 untuk menulis ke dalam *SD card*. Nilai yang ditulis adalah bilangan hexa mulai 0x00 sampai 0xFF secara berurutan ke dalam suatu alamat tertentu di dalam *SD card* yang kemudian diulangi lagi hingga mengisi sebanyak satu *sector* atau 512 byte
b. Menghidupkan sistem yang telah dipasang *SD card*
c. Melepas *SD card* kemudian dibaca melalui komputer dengan menggunakan *software* WinHex

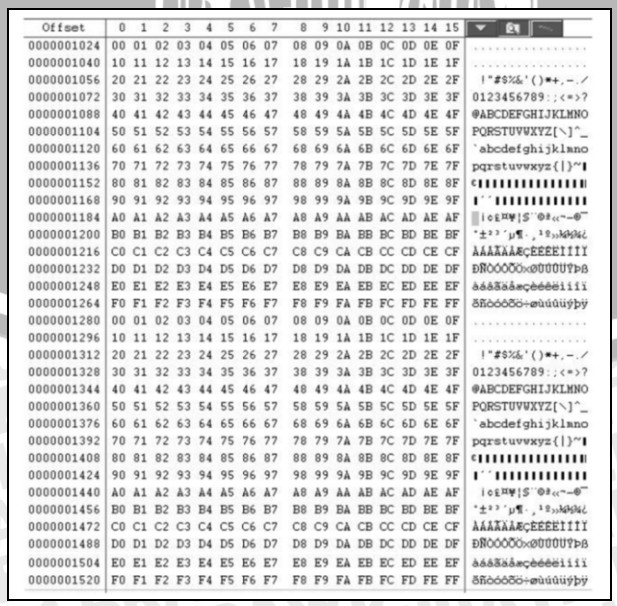
5.3.1.4 Hasil Pengujian dan Analisis

Rangkaian pengujian penulisan data ke dalam *SD card* menggunakan mikrokontroler R8C/13 ditunjukkan dalam Gambar 5.8



Gambar 5.8 Rangkaian pengujian penulisan data ke dalam *SD card* menggunakan mikrokontroler R8C/13

Dalam pengujian ini, mikrokontroler telah dapat menulis ke dalam *SD card*. Dengan menggunakan *software* WinHex, ditunjukkan pada offset 1024 sampai 1520+15, sebanyak satu *sector* penuh, yaitu 512 byte telah berisi nilai hexa 0x00 sampai 0xFF yang dilakukan sebanyak dua kali penulisan. Gambar 5.9 menunjukkan tampilan pembacaan *SD card* menggunakan *software* WinHex.



Gambar 5.9 Tampilan Pembacaan *SD card* menggunakan *software* WinHex

5.3.2 Pengujian Baca *SD card*

5.3.2.1 Tujuan

Pengujian ini dilakukan untuk menganalisis apakah mikrokontroler dapat membaca data dari dalam *SD card* dengan baik

5.3.2.2 Peralatan Pengujian

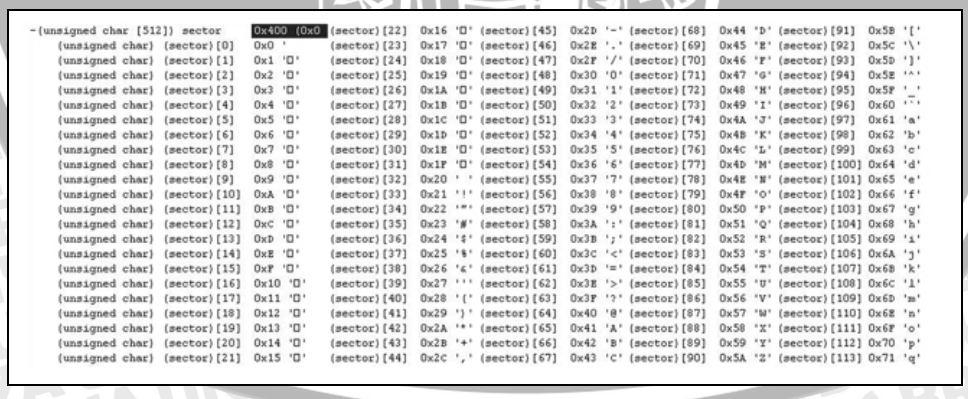
a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000
b. *SD card*
c. *Software* HEW
d. *Software* WinHex

5.3.2.3 Prosedur Pengujian

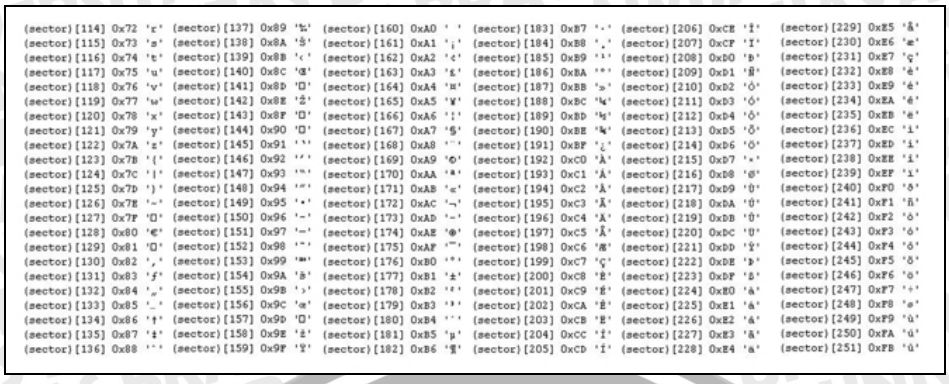
a. Membuat program mikrokontroler Renesas R8C/13 untuk membaca data dari *SD card*
b. Menulis data ke dalam *SD card*
c. Menghidupkan sistem dan mengamati isi variabel hasil pembacaan *SD card* melalui program HEW

5.3.2.4 Hasil Pengujian dan Analisis

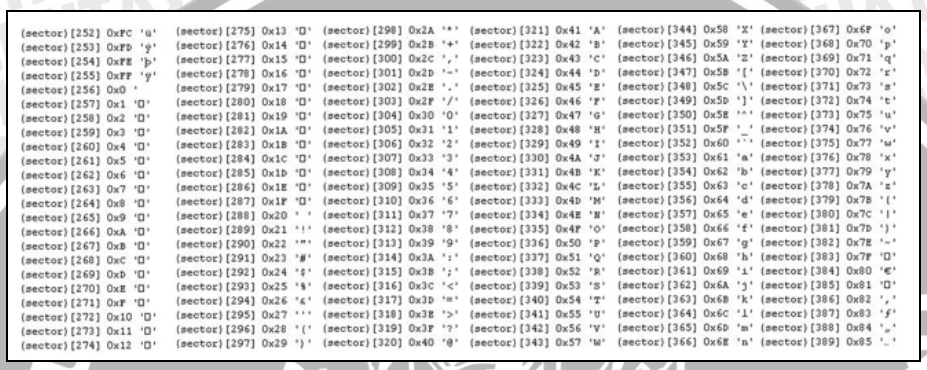
Dalam pengujian ini, mikrokontroler telah dapat membaca data dari *SD card.* Dengan menggunakan *software* HEW, isi variabel hasil pembacaan *SD card* yaitu sector[512] telah berisi nilai hexa 0x00 sampai 0xFF yang dilakukan sebanyak dua kali penulisan. Gambar 5.10 sampai 5.13 menunjukkan tampilan pembacaan *SD card* menggunakan *software* HEW.



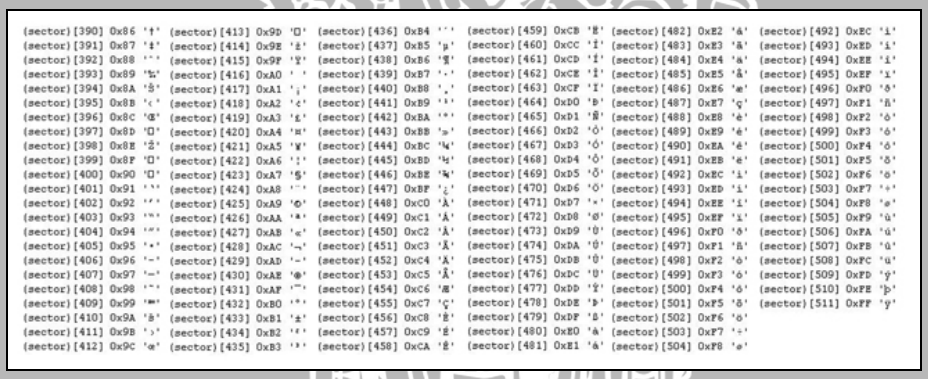
Gambar 5.10 Tampilan Pembacaan *SD card* menggunakan *software* HEW



Gambar 5.11 Tampilan Pembacaan *SD card* menggunakan *software* HEW (lanjutan 1)



Gambar 5.12 Tampilan Pembacaan *SD card* menggunakan *software* HEW (lanjutan 2)



Gambar 5.13 Tampilan Pembacaan *SD card* menggunakan *software* HEW (lanjutan 3)

5.4 Pengujian Modul LCD

5.4.1 Tujuan

Pengujian modul LCD dilakukan untuk menganalisis apakah LCD dapat menampilkan karakter dengan benar

5.4.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000

b. Modul LCD 2×16

c. *Software* HEW

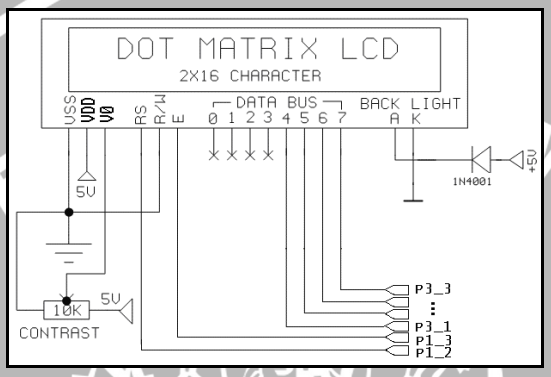
5.4.3 Prosedur Pengujian

a. Membuat program mikrokontroler Renesas R8C/13 untuk menulis beberapa karakter ke LCD

b. Menghidupkan sistem dan mengamati LCD

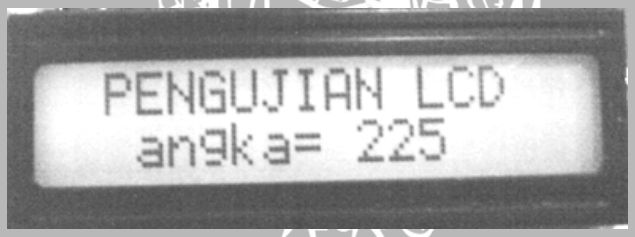
5.4.4 Hasil Pengujian dan Analisis

Rangkaian pengujian penulisan karakter dalam LCD menggunakan mikrokontroler R8C/13 ditunjukkan dalam Gambar 5.14. P1_1, P1_2, P3_1 hingga P3_3 dihubungkan dengan pin mikrokontroler.



Gambar 5.14 Rangkaian pengujian penulisan karakter dalam LCD menggunakan mikrokontroler R8C/13

Dalam pengujian ini, LCD telah dapat menampilkan beberapa karakter yang dikirim oleh mikrokontroler. Gambar 5.15 menunjukkan tampilan LCD dengan beberapa karakter huruf dan angka.



Gambar 5.15 Tampilan LCD 2×16

5.5 Pengujian Rangkaian *Bidirectional Level shifter* 3 V-5 V

5.5.1 Tujuan

Pengujian *bidirectional level shifter* dilakukan untuk menganalisis apakah rangkaian dapat mengubah level tegangan logika 3,3 V menjadi 5 V atau sebaliknya dengan benar

5.5.2 Peralatan Pengujian

a. Rangkaian *bidirectional level shifter*

b. *Function generator*

c. Osiloskop digital

5.5.3 Prosedur Pengujian

a. Membuat sinyal pulsa diskrit dengan amplitudo logika high sebesar sekitar 3 V dengan frekuensi sekitar 100 Hz dengan mempergunakan *function generator*

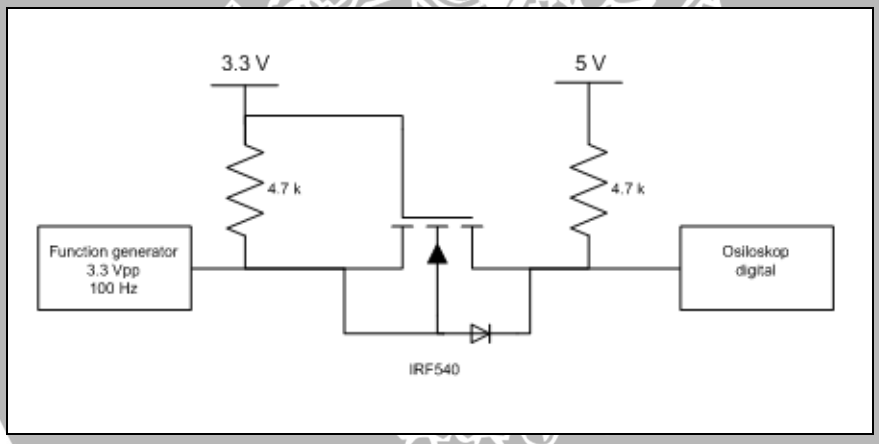
b. Sinyal tersebut dihubungkan ke masukan rangkaian *bidirectional level shifter*

c. Sinyal keluaran rangkaian *bidirectional level shifter* diamati dengan osiloskop digital

d. Dengan cara yang sama, diulangi dengan sinyal pulsa diskrit masukan sebesar 5 Vpp dengan frekuensi sekitar 100 Hz

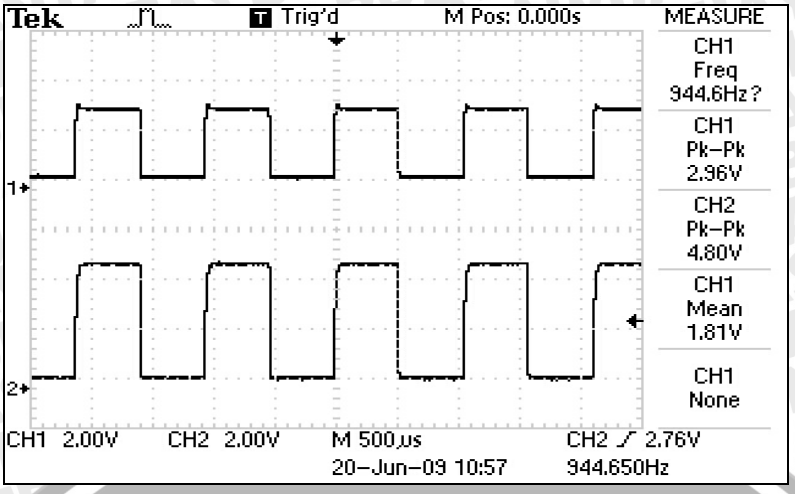
5.5.4 Hasil Pengujian dan Analisis

Pengujian dilakukan dalam dua tahap. Pertama, rangkaian *bidirectional level shifter* diberi masukan sinyal sekitar 3 Vpp dan diamati bentuk sinyal keluarannya. Kedua, rangkaian *bidirectional level shifter* diberi masukan sinyal sekitar 5 Vpp dan diamati bentuk sinyal keluarannya. Rangkaian untuk pengujian pertama *bidirectional level shifter* ditunjukkan di dalam Gambar 5.16.



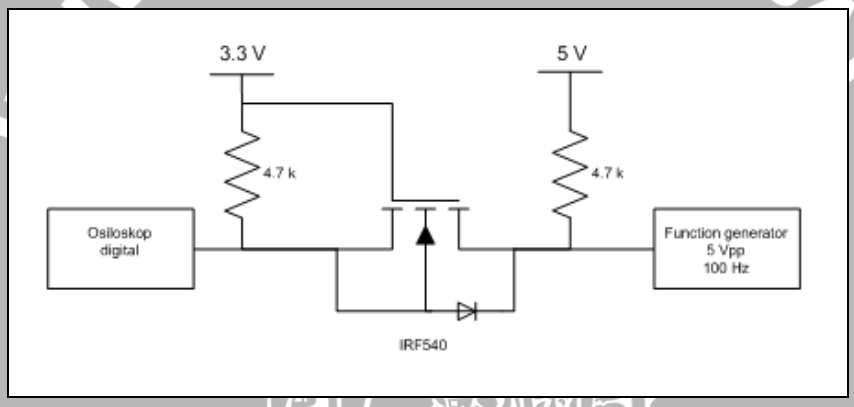
Gambar 5.16 Rangkaian untuk pengujian pertama *bidirectional level shifter*

Dalam pengujian pertama, rangkaian *bidirectional level shifter* dapat mengubah sinyal pulsa masukan sebesar sekitar 3 Vpp dan frekuensi sekitar 100 Hz menjadi sinyal dengan amplitudo sekitar 5 V. Gambar 5.17 menunjukkan gambar sinyal masukan dan keluaran yang terbaca melalui osiloskop. CH1 mewakili sinyal masukan dan CH2 mewakili sinyal keluaran. Di dalam Gambar 5.17 terlihat bahwa CH1 osiloskop sebesar 2,96 V, CH2 osiloskop sebesar 4,8 V dan freq sebesar 944,6 Hz.



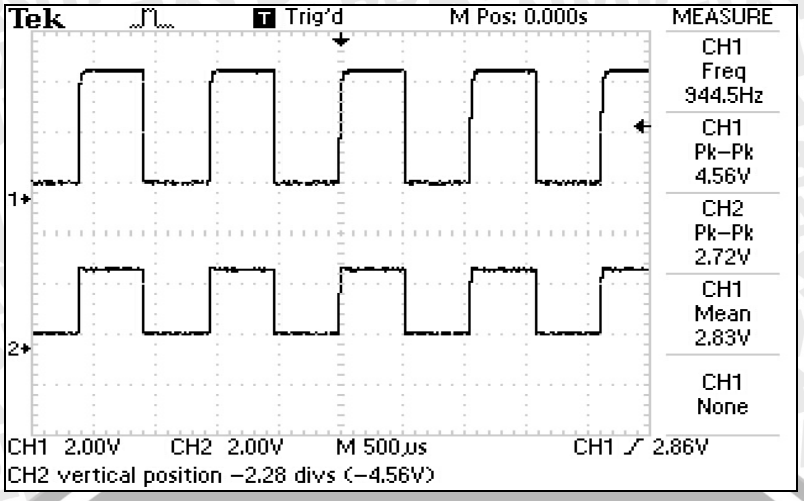
Gambar 5.17 Tampilan osiloskop dari sinyal masukan dan keluaran rangkaian

Rangkaian untuk pengujian kedua *bidirectional level shifter* ditunjukkan di dalam Gambar 5.18.



Gambar 5.18 Rangkaian untuk pengujian kedua *bidirectional level shifter*

Dalam pengujian kedua, rangkaian *bidirectional level shifter* dapat mengubah sinyal pulsa masukan sebesar sekitar 5 Vpp dan frekuensi sekitar 100 Hz menjadi sinyal dengan amplitudo sekitar 3 V. Gambar 5.19 menunjukkan gambar sinyal masukan dan keluaran yang terbaca melalui osiloskop. CH1 mewakili sinyal masukan dan CH2 mewakili sinyal keluaran. Di dalam Gambar 5.19 terlihat bahwa CH1 osiloskop sebesar 4,56 V, CH2 osiloskop sebesar 2,72 V dan freq sebesar 944,5 Hz.



Gambar 5.19 Tampilan osiloskop dari sinyal masukan dan keluaran rangkaian

5.6 Pengujian RTC

5.6.1 Tujuan

Pengujian RTC dilakukan untuk menganalisis apakah RTC dapat memberikan informasi waktu yang benar

5.6.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000

b. Rangkaian sistem RTC DS1307

c. Modul LCD 2×16

d. *Software* HEW

5.6.3 Prosedur Pengujian

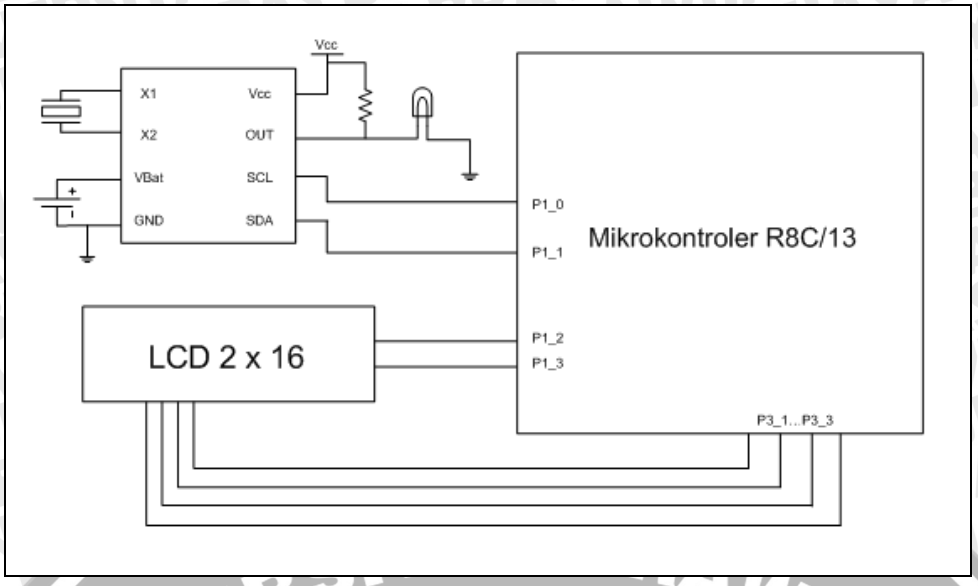
a. Membuat program mikrokontroler Renesas R8C/13 untuk mengisi register pewaktuan ke dalam RTC sekaligus untuk membaca kembali register tersebut dan ditampilkan melalui LCD 2×16 dengan format jam:menit:detik dan tanggal/bulan/tahun yang dilakukan secara terus menerus

b. Menghidupkan sistem yang terdiri atas mikrokontroler R8C/13, RTC dan LCD.

c. Mengamati tampilan LCD

5.6.4 Hasil Pengujian dan Analisis

Rangkaian pengujian RTC ditunjukkan dalam Gambar 5.20. LCD digunakan untuk menampilkan register pewaktuan didalam RTC yang meliputi tanggal, bulan, tahun, jam, menit dan detik.



Gambar 5.20 Rangkaian pengujian RTC

Dalam pengujian ini, Sistem pewaktuan telah berjalan dengan benar. Register RTC telah dapat ditulis dan dibaca kembali oleh mikrokontroler yang untuk selanjutnya ditampilkan melalui LCD. Gambar 5.21 menunjukkan sistem pewaktuan berbasis RTC yang ditampilkan melalui LCD.

Gambar 5.21 Sistem pewaktuan berbasis RTC yang ditampilkan melalui LCD

5.7 Pengujian Sistem Keseluruhan

5.7.1 Tujuan

Pengujian sistem secara keseluruhan dilakukan untuk menganalisis apakah sistem dapat bekerja dengan benar sesuai dengan yang telah dirancang sebelumnya, yakni merekam data analog secara periodik ke dalam *SD card* dalam format file teks sehingga selanjutnya dapat dianalisis menggunakan komputer.

5.6.2 Peralatan Pengujian

a. Mikrokontroler Renesas R8C/13 berupa modul HRS8000
b. Rangkaian sistem RTC DS1307
c. Modul LCD 2×16
d. Rangkaian *Bidirectional Level Shifter*
e. *SD card*

f. Rangkaian *push button*

g. Catu daya Adaptor

h. *Software* HEW

5.6.3 Prosedur Pengujian

a. Merangkai seluruh unit sistem menjadi satu kesatuan seperti yang telah ditunjukkan di dalam diagram blok sistem

b. Memprogram mikrokontroler R8C/13 dengan *software* yang telah dirancang untuk dapat membaca *push button*, ADC, menulis dan membaca register RTC serta berkomunikasi dengan *SD card*.

c. Memberikan sinyal analog masukan berupa sinyal suhu dengan sensor LM35 serta sinyal tegangan potensiometer yang diubah secara manual

d. Menghubungkan sistem dengan catu daya dan menghidupkan sistem

e. Mengamati sistem selama selang periode tertentu

f. Mematikan sistem kemudian mengambil *SD card* untuk selanjutnya dianalisis menggunakan *software* komputer

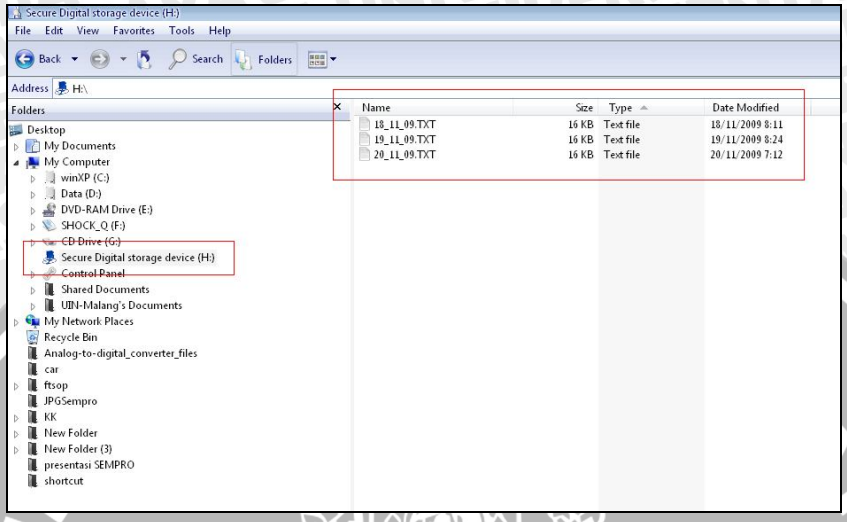
5.6.4 Hasil Pengujian dan Analisis

Pada saat sistem dijalankan, sistem telah bekerja dengan benar. Hal ini ditunjukkan oleh tampilan LCD pada sistem yang ditunjukkan dalam Gambar 5.22. Seperti yang ditunjukkan, sistem telah dapat menjalankan sistem pewaktuan dengan benar sesuai dengan yang telah dirancang. Sistem juga telah dapat membaca sinyal masukan ADC yang berjumlah dua kanal. Sinyal masukan analog untuk ADC kanal pertama berasal dari sinyal tegangan potensiometer yang dapat diubah-ubah secara manual dan sinyal analog kanal kedua berasal dari sinyal suhu yang disediakan rangkaian sensor LM35. Gambar 5.22 menunjukkan sistem dijalankan dengan sampling data setiap dua menit sekali. Data ADC yang terbaca pada kanal A adalah 0x213 dan data yang terbaca pada kanal B adalah 0x124. Sistem dijalankan pada tanggal 5 bulan 10 pada pukul 20.00 lebih 14 detik.

Gambar 5.22 Tampilan LCD ketika sistem dijalankan

Pengujian selanjutnya dilakukan dalam jangka waktu tiga hari untuk mengecek apakah file dapat tercipta sesuai dengan jumlah hari. Setelah tiga hari, sistem dimatikan

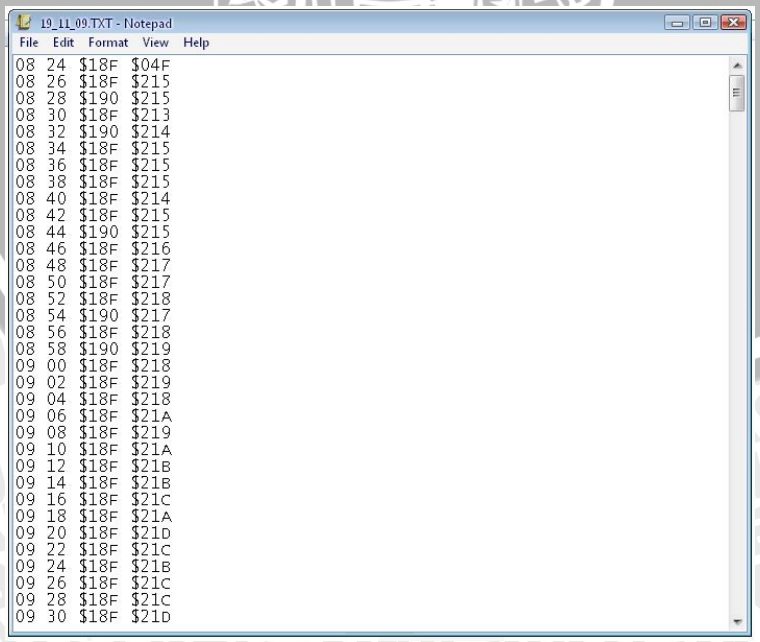
dan *SD card* dilepas untuk selanjutnya diamati melalui komputer. Dengan menggunakan Windows Explorer, file yang terdapat di dalam *SD card* dapat diakses. Gambar 5.23 menunjukkan 3 file teks hasil rekaman yang ditampilkan melalui Windows Explorer.



Gambar 5.23 File teks hasil rekaman yang ditampilkan melalui Windows Explorer

Karena sistem dijalankan selama 3 hari, maka file teks yang muncul juga berjumlah tiga buah. Dengan penamaan file yang mengacu pada tanggal file dibuat, maka dapat disimpulkan bahwa file dibuat pada tanggal 18, 19 dan 20 bulan 11.

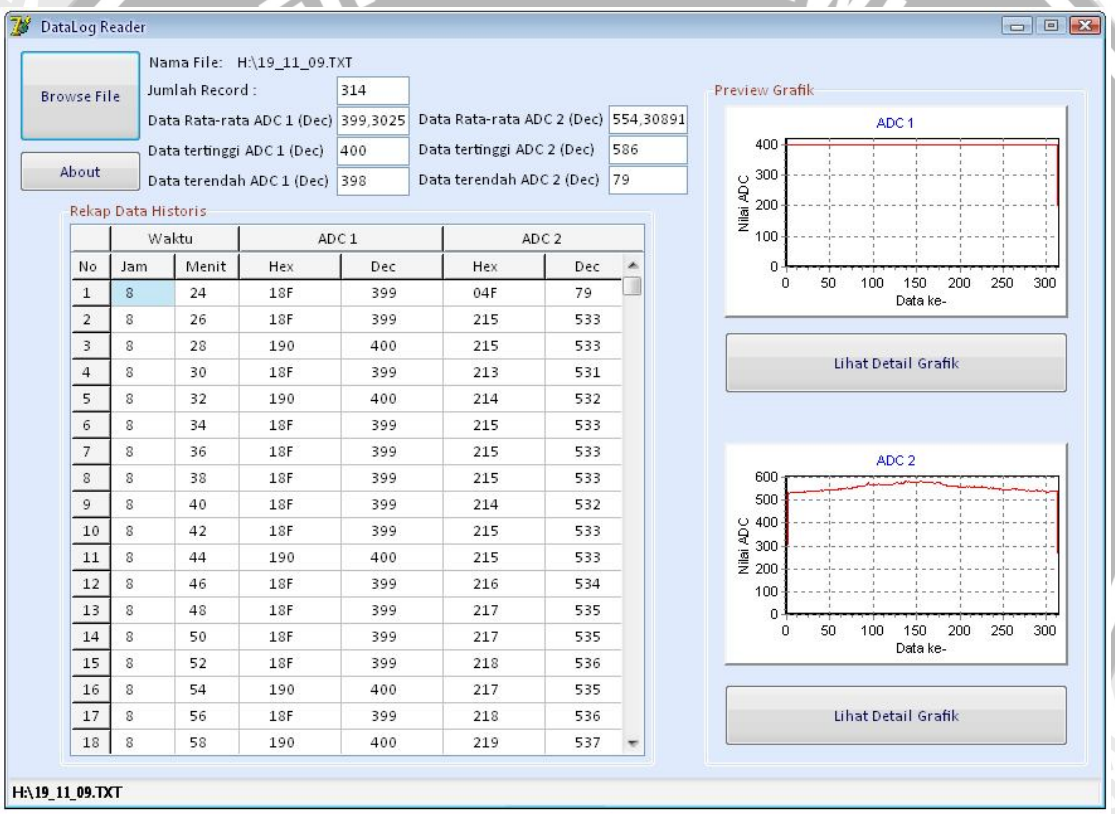
File teks dapat diakses langsung menggunakan *software* teks editor semacam notepad. Gambar 5.24 menunjukkan file yang dibuat pada tanggal 19 bulan 11 tahun 2009 diakses menggunakan notepad.



Gambar 5.24 File teks yang berisi rekaman data dengan sampling dua menit

Data yang tertulis pada notepad mengandung beberapa informasi antara lain jam dan menit ketika sampling data dilakukan, serta data ADC yang terbaca pada kanal 1 dan 2. Kolom pertama menunjukkan jam sedangkan kolom kedua menunjukkan menit. Kolom ketiga dan keempat secara berurutan adalah nilai ADC yang terbaca pada kanal 1 dan 2. Karakter '$' sebelum nilai ADC 1 dan 2 menunjukkan bahwa data tersebut adalah data dalam bilangan hexadesimal.

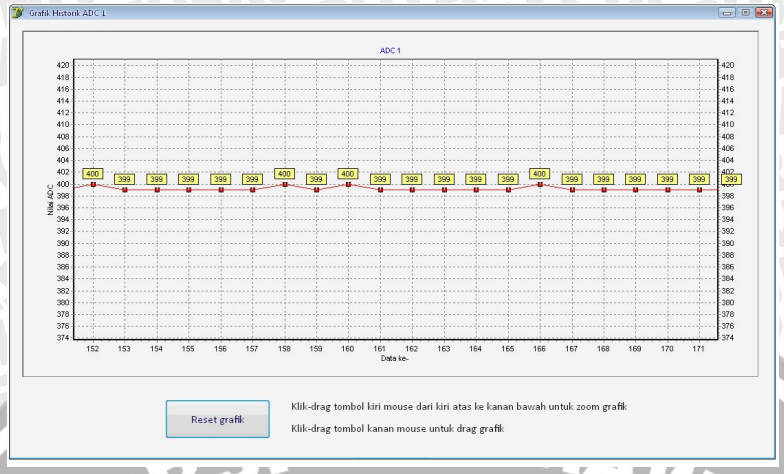
Untuk mempermudah pembacaan data, file teks tersebut dibuka menggunakan *software* datalog *reader* yang telah dibuat sebelumnya. *Software* ini dapat menunjukkan rekaman data dalam bentuk tabel dan grafik sehingga mempermudah analisis data. Gambar 5.25 menunjukkan file teks yang diakses menggunakan *software* datalog *reader*.



Gambar 5.25 File teks yang diakses menggunakan *software* untuk menampilkan tabel dan grafik

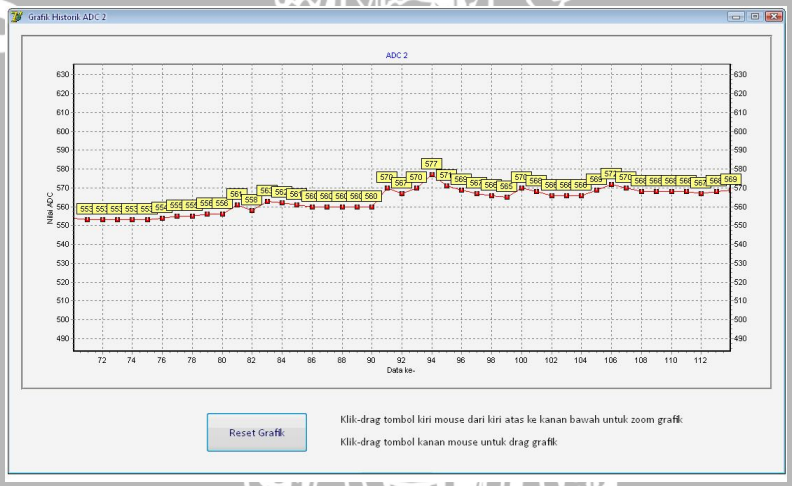
Tombol '*Browse*' digunakan untuk memilih file teks yang akan diakses. Tabel data berisi informasi-informasi di dalam file teks yang telah dipisahkan menurut kolom bersesuaian. Di sebelah kanan terdapat *preview* grafik sinyal analog dari ADC kanal 1 dan 2. Terdapat tombol di bawah *preview* untuk melihat grafik secara lebih detail. Grafik yang ditampilkan secara detail dapat dilakukan *zooming* dan *panning*. *Zoom* berarti melihat grafik dalam ukuran lebih besar, sedangkan *pan* berarti menggeser-geser

grafik. Gambar 5.26 menunjukkan grafik nilai masukan ADC 1 yang ditampilkan secara detail.



Gambar 5.26 Grafik nilai masukan ADC 1

Gambar 5.27 menunjukkan grafik nilai masukan ADC 2 yang ditampilkan secara detail.



Gambar 5.27 Grafik nilai masukan ADC 2

Dari pengujian sistem secara keseluruhan yang telah dilakukan, dapat disimpulkan bahwa sistem telah bekerja dengan benar dan sesuai dengan yang telah dirancang.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil pengujian tiap bagian dan keseluruhan sistem yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut:

1. Sistem pewaktuan eksternal mikrokontroler dengan menggunakan RTC DS1307 mudah untuk diimplementasikan dalam sistem karena memiliki register informasi waktu yang meliputi detik, menit, jam, hari, tanggal, bulan dan tahun dalam format bilangan BCD yang dapat diakses dengan mudah menggunakan protokol komunikasi I2C. Hasil pengujian sistem pewaktuan dengan RTC DS1307 menunjukkan bahwa IC telah dapat memberikan data informasi waktu dengan benar melalui penampil LCD.
2. Rangkaian *bidirectional level shifter* yang tersusun atas rangkaian FET sebagai *switch* dapat mengubah level tegangan protokol komunikasi antara mikrokontroler dan RTC DS1307 yang masing-masing berada pada level tegangan logika tinggi yang berbeda, yakni +3,3 V dan +5 V. Tingkat kesalahan yang terjadi adalah sekitar 8% sampai 12%.
3. *SD card* menyediakan protokol komunikasi sekunder berupa SPI dengan melakukan prosedur inisialisasi tertentu dan clock maksimum saat inisialisasi sebesar 400 kHz. Hasil pengujian *SD card* menunjukkan bahwa *SD card* telah dapat dibaca dan ditulis menggunakan mikrokontroler dengan antarmuka SPI.
4. Dengan mempergunakan format penyimpanan data dalam *SD card* yang mematuhi aturan penulisan *filesystem* FAT16 maka data sinyal analog yang telah dikonversi menjadi data digital yang telah tersimpan di dalam *SD card* dapat terbaca melalui komputer dalam bentuk sebuah file.
5. Hasil pengujian sistem secara keseluruhan menunjukkan bahwa sistem data logger telah dapat mengambil sampel data sinyal analog melalui dua kanal ADC secara periodik dalam selang waktu tertentu yang kemudian disimpan ke dalam *SD card* untuk selanjutnya dibaca menggunakan komputer.

6.2 Saran

Saran-saran dalam pengimplementasian maupun peningkatan unjuk kerja sistem dalam penelitian ini dapat diuraikan sebagai berikut:

1. Saat ini sistem dicatu oleh adaptor yang terhubung dengan sumber listrik jala-jala PLN. Oleh karena itu, disarankan adanya pengembangan catu daya yang lebih bersifat portabel sehingga sistem tidak bergantung dengan sumber listrik jala-jala PLN
2. Sistem dapat dikembangkan dengan penambahan jumlah kanal masukan sinyal ADC yang dapat direkam sekaligus memberikan pilihan kecepatan sampling data yang lebih bervariasi
3. Sistem dapat dikembangkan lebih lanjut dengan penggunaan *filesystem SD card* yang lebih umum semisal FAT32 atau NTFS. Selain itu, sistem yang saat ini hanya dapat menulis file di root entri utama, dapat dikembangkan dengan prosedur penulisan ke dalam bentuk folder-folder sehingga dapat memaksimalkan kapasitas *SD card* sebagai unit penyimpanan

DAFTAR PUSTAKA

AccessData. 2008. *Introduction to DOS and FAT*. AccessData Corporation. *http:// www.accessdata.com/downloads/media/Introduction to DOS & FAT 7-29-08.pdf.* Diakses 23 Maret 2009

Allen, R. L. dan Mills, D. W. 2004. *Signal Analysis: Time, Frequency, Scale and Structure*. IEEE Press.

Cantu, Marco. 2003. *Mastering Delphi 7*. Sybex Inc.

Carrier, Brian. 2005. *File System Forensic Analysis.* Addison Wesley Professional

Dallas. 2000. *DS1307/DS1308 64 X 8 Serial Real Time Clock.* Dallas Semiconductor. *http:/www. pdf1. alldatasheet. Com / datasheet-pdf /view / 58481 /DALLAS/DS1307.html*. Diakses 16 Januari 2009

ESAcademy. 2000. *The I2C protocol*. *http://www.esacademy.com /faq /i2c /general /i2cproto.htm*. Diakses 15 Januari 2009

Heybruck, William F. 2007. *An Introduction to FAT16/FAT32 Filesystem*. Hitachi Global Storage Technology

Hitachi. 2006. *LMB162 Specification.* http://*www.datasheetcatalog.com*. Diakses 15 Januari 2009

Mutohar, Amin. 2008. *Komunikasi Data SPI pada Mikrokontroler*. ITB

Philips. 2003. *I2C Manual Application Note*. Philips Semiconductor. *http://www.nxp.com/acrobat_download/applicationnotes/AN10216_1.pdf*. Diakses 23 Maret 2009

Renesas. 2005. *R8C/13 Group Datasheet Rev 1.10.* Renesas Technology

Renesas. 2005. *R8C/13 Group Hardware Manual Rev 1.10.* Renesas Technology

Renesas. 2005. *R8C/13 Group Software Manual Rev 1.10.* Renesas Technology

SanDisk. 2003. *SanDisk Secure Digital Card Product Manual.* SanDisk Corporation. *http:// www.convict.lu/pdf/ProdManualSDCardv1.9.pdf*. Diakses 10 Januari 2009

Sulistiyanto, Nanang. 2008. Pemrograman Mikrokontroler R8C/13. Jakarta: PT Elex Media Komputindo

Texas Instrument. 2007. *I2C Selection Guide*. Texas Instruments. *http:// focus.ti.com/pdfs/logic/i2cselguide.pdf*. Diakses pada 23 Maret 2009

Thompson, Ed. t.t. *I2C Bus Technical Overview.* Micro Computer Control.

Transcend. 2009. *TS1G-2GSDG Secure Digital Card*. Transcend Information Inc.

LAMPIRAN

LAMPIRAN I

FOTO ALAT

Gambar 1 Alat tampak depan

Gambar 2 Tempat SD card dan port tegangan referensi sebesar 3,44 V untuk sistem eksternal

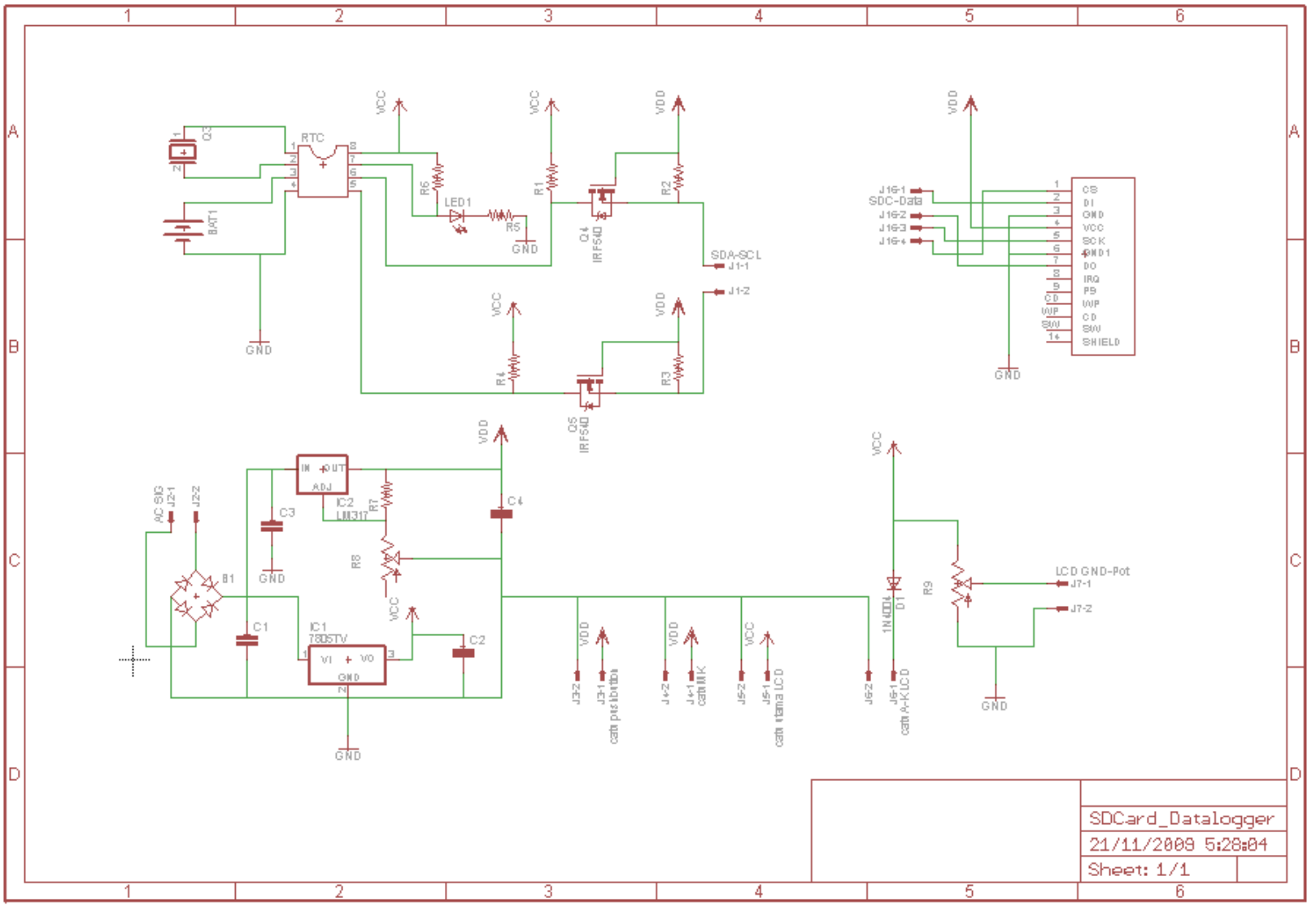
Gambar 3 Port catu daya adaptor dan saklar utama sistem

Gambar 4 Port ADC 1 dan ADC 2

Gambar 5 Rangkaian di dalam sistem

LAMPIRAN II

GAMBAR RANGKAIAN


RTC
BAT1
LED1
R1
R2
R3
R4
R5
R6
Q3
Q4
Q5
IRF540
VCC
VDD
GND
SDA-SCL
J1-1
J1-2
SDC-Data
J16-1
J16-2
J16-3
J16-4
CS
DI
GND
VCC
SCK
GND1
DO
IRQ
P9
WP
CD
SW
SHIELD
AC SIG
J2-1
J2-2
B1
C1
C2
C3
C4
IC1
7805TV
IC2
LM317
ADJ
R7
R8
J3-2
J3-1
J4-2
J4-1
J5-2
J5-1
J6-2
J6-1
D1
1N4004
R9
LCD GND-Pot
J7-1
J7-2
SDCard_Datalogger
21/11/2009 5:28:04
Sheet: 1/1

LAMPIRAN III

LISTING PROGRAM

MIKROKONTROLER RENESAS R8C/13

PROGRAM UTAMA

```
/*************************************************************
        Tempat deklarasi include
*************************************************************/
#include    "sfr_r813.h"                    /* Definition of the R8C/13 SFR */
#include    "RTC.h"
#include    "IIC_bus.h"
#include    "seting_port.h"
#include    "seting_adc.h"
#include    "seting_interupsi.h"
#include    "seting_timer_x.h"
#include    "seting_clock.h"
#include    "lcd_4_bit.h"
#include    "lcd_4_bit.c"
#include    "command.h"
#include    "FAT.h"
#include    "command.h"
#include    "mmc.h"

extern unsigned char tglskrg=0x00;
extern unsigned int clust_x;
unsigned int adc1;
unsigned int adc2;
extern unsigned char sector[512];
extern unsigned long root_directory;
char sampling;//bwat kec sampling
unsigned char menitskrg; //bwat ngecek apakah sekarang saat sampling

void tunggu_adc_stabil()
{
        char k=200;
        while(k!=0)k--;
}

/*************************************************************************
*       Function : main()
*       program section
*************************************************************************/
void main()
        {
        char ada,i;
        unsigned long sect_clustkosong;

        CLOCK_SKALA_8;
        CLOCK_INTERNAL_F_TINGGI;
        i=0;
//      for(i=0;i<5;i++)SetJam();               //set jam di rtc.c
        for(i=0;i<5;i++)ReadTime();
        CLOCK_SKALA_1;
        CLOCK_EKSTERNAL;
        pd1=0x00;
        pd1=0xff;
        p1=0xff;
        ADC_KONVERSI_TAK_KONTINYU;
        ADC_10_BIT;
        ADC_CLOCK_F2;
        ADC_PENCUPLIK_AKTIF;
        ADC_ACUAN_AKTIF;
        sampling=0x00;
        tglskrg=date;
        menitskrg=minute;
        lcd_inisialisasi();
        lcd_hidup_tanpa_kursor();
        lcd_hapus_layar();
        pd0_4=0;
        pd0_5=0;
        pd0_6=0;
        pd0_7=0;
        lcd_posisi_baris_1(1);
        lcd_kirim("<MENU SAMPLING>");
        lcd_posisi_baris_2(1);
        lcd_kirim("Tekan tomboi...");
        //perintah pushbutton,,dikasi while byar programe nunggu
        while((p0_4)||(p0_5)||(p0_6)||(p0_7))
        {
                //jika pushbutton1 ditekan sampling=2 menit
                if(p0_4==0){sampling=0x02;break;}
                //jika pushbutton1 ditekan sampling=5 menit
                else if(p0_5==0){sampling=0x05;break;}
                //jika pushbutton1 ditekan sampling=10 menit
                else if(p0_6==0){sampling=0x10;break;}
                //jika pushbutton1 ditekan sampling=30 menit
                else if(p0_7==0){sampling=0x30;break;}
        }

        lcd_hapus_layar();
        lcd_posisi_baris_1(2);
        lcd_kirim("Please Wait");
        lcd_posisi_baris_2(2);
        lcd_kirim("LOADING....");

        init_SPI();
        while(init_MMC()!=0x00);
        u0c0    = 0x88;                         // clock = f1SIO/(2*(4+1) = 20/10 = 2 MHz
        u0brg   = 0x04;
        while(offset_partisi()== 0x00);
        read_file_system_info();
        hit_jum_clust_kosong();

        /*script untuk ngecek apakah dah ada file pada tanggal sekarang*/
        read_block(root_directory);
        ada=0;
        for(i=0;i<16;i++)
```

```
{               if((sector[i*32]!=0x00)&&(sector[i*32]!=0xE5))
                ada++;
                else if ((sector[i*32]==0x00)||(sector[i*32]==0xE5))
                ada=ada;
}
if(ada==0)
{               //jika tidak ada file kosongkan fisik 2
                sector[0]=0x00;
                sector[1]=0x00;
                write_block_fisik(2);
}
                //jika ada file,langsung baca sector 2 fisik bwat nentuin alamat cluster lanjutan
                //yang dimaksud adalah alamat file terakhir ktika system dimatiin
read_block_fisik(2);
                //isinya cuma 2 byte:byte 1 berisi tgl file trakhir, byte 2 berisi alamat clusternya
 if(sector[0]==date)
{
                clust_x=sector[2]<<8;
                clust_x=sector[1]+clust_x;
}
Else            //jika sudah ganti hari
 {
                tulis_data_awal();
                //buat cluster,kosongkan cluster
                tulis_root(root_directory);             //bwat nama file baru di root
                updatefat();
}
/*selesai*/
CLOCK_SKALA_1;
CLOCK_EKSTERNAL;
lcd_hapus_layar();
lcd_posisi_baris_2(2);
lcd_kirim_karakter('/');
lcd_posisi_baris_2(5);
lcd_kirim_karakter('/');
lcd_posisi_baris_2(8);
lcd_kirim_karakter(':');
lcd_posisi_baris_2(11);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(0);
lcd_kirim_karakter('#');
lcd_posisi_baris_1(3);
lcd_kirim_karakter('/');
lcd_posisi_baris_1(4);
lcd_kirim_karakter('A');
lcd_posisi_baris_1(5);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(9);
lcd_kirim_karakter('/');
lcd_posisi_baris_1(10);
lcd_kirim_karakter('B');
lcd_posisi_baris_1(11);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(1);
lcd_kirim_hex(sampling);
//sampling awal ketika system baru dinyalain

TIMER_X_PENGUKUR_SELANG_WAKTU;
TIMER_X_CLOCK_F32;              //
prex=249;                       //  50 Hz
tx=49;                          //
ADC_KANAL6;
tunggu_adc_stabil();
adc1=0;
for(i=0;i<25;i++)
{               ADC_KONVERSI_AKTIF;
                adc1=adc1+ad;
                TIMER_X_PENCACAHAN_AKTIF;
                while(txund == 0){};
                TIMER_X_PENCACAHAN_MATI; txund=0;
}
adc1=adc1/i;
lcd_posisi_baris_1(6);
lcd_kirim_hex_3(adc1);
ADC_KANAL4;
tunggu_adc_stabil();
adc2=0;
for(i=0;i<25;i++)
{               ADC_KONVERSI_AKTIF;
                adc2=adc2+ad;
                TIMER_X_PENCACAHAN_AKTIF;
                while(txund == 0){};
                TIMER_X_PENCACAHAN_MATI; txund=0;
}
adc2=adc2/i;
lcd_posisi_baris_1(12);
lcd_kirim_hex_3(adc2);
//tulis data ke file yang sudah dideklarasi di awal tadi posisi clusternya
//byar gak sampling lagi
menitskrg=menitskrg+sampling;
tulis_data();
while(1)
{
                //konversi menit yg terbaca,,kan bil BCD
                if (menitskrg>0x59)menitskrg=menitskrg+0x06;
                if(menitskrg>=0x60)menitskrg=menitskrg-0x60;

        if(((menitskrg>0x09)&&(menitskrg<0x10))||((menitskrg>0x19)&&(menitskrg<0x20))||((menitskr
g>0x29)&&(menitskrg<0x30))||((menitskrg>0x39)&&(menitskrg<0x40))||((menitskrg>0x49)&&(menitskrg<
0x50)))menitskrg=menitskrg+0x06;
                else menitskrg=menitskrg;
                CLOCK_SKALA_8;
                CLOCK_INTERNAL_F_TINGGI;
                ReadTime();
```

```
CLOCK_SKALA_1;
CLOCK_EKSTERNAL;

lcd_posisi_baris_2(2);
lcd_kirim_karakter('/');
lcd_posisi_baris_2(5);
lcd_kirim_karakter(' ');
lcd_posisi_baris_2(8);
lcd_kirim_karakter(':');
lcd_posisi_baris_2(11);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(0);
lcd_kirim_karakter('#');
lcd_posisi_baris_1(3);
lcd_kirim_karakter(' ');
lcd_posisi_baris_1(4);
lcd_kirim_karakter('A');
lcd_posisi_baris_1(5);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(9);
lcd_kirim_karakter(' ');
lcd_posisi_baris_1(10);
lcd_kirim_karakter('B');
lcd_posisi_baris_1(11);
lcd_kirim_karakter(':');
lcd_posisi_baris_1(1);
lcd_kirim_hex(sampling);
lcd_posisi_baris_2(0);
lcd_kirim_hex(date);
lcd_posisi_baris_2(3);
lcd_kirim_hex(month);
lcd_posisi_baris_2(6);
lcd_kirim_hex(hour);
lcd_posisi_baris_2(9);
lcd_kirim_hex(minute);
lcd_posisi_baris_2(12);
lcd_kirim_hex(second);
if((minute==menitskrg))
{
        ADC_KANAL6;
        tunggu_adc_stabil();
        adc1=0;
        for(i=0;i<25;i++)
        {       ADC_KONVERSI_AKTIF;
                adc1=adc1+ad;
                TIMER_X_PENCACAHAN_AKTIF;
                while(txund == 0){};
                TIMER_X_PENCACAHAN_MATI; txund=0;
        }

        adc1=adc1/i;
        lcd_posisi_baris_1(6);
        lcd_kirim_hex_3(adc1);
        ADC_KANAL4;
        tunggu_adc_stabil();
        adc2=0;
        for(i=0;i<25;i++)
        {       ADC_KONVERSI_AKTIF;
                adc2=adc2+ad;
                TIMER_X_PENCACAHAN_AKTIF;
                while(txund == 0){};
                TIMER_X_PENCACAHAN_MATI; txund=0;
        }

        adc2=adc2/i;
        lcd_posisi_baris_1(12);
        lcd_kirim_hex_3(adc2);
        //biar ga trus2an

        menitskrg=menitskrg+sampling;
         //deteksi apakah ganti tanggal,,
         //kalo belum ganti tanggal
        if(date==tglskrg)
                        tulis_data();                           //tulis ad
         //kalo udah ganti tgl
        else{
                        tulis_data_awal();
                       //cari cluster baru utk file baru
                       tulis_data();
                        tulis_root(root_directory);//bwat file baru di root
                       updatefat();                    //update fat
                        tglskrg=date;  //sinkronisasi
                       }


                                }
                                else

        ADC_KONVERSI_MATI;
                        }
        }
```

SUBRUTIN LCD

```
void lcd_tunggu(void)
{
  int k;
  for(k = 0; k < 1500; k++);
}

void lcd_tunggu_lama(void)
{
  long k;
  for(k = 0; k < 50000; k++);
}

void lcd_kirim_perintah_8_bit(char rs, char data)
{
```

```
  union byte_def data_terima;
  data_terima.byte = data;
  LCD_RS = rs;
  LCD_DIR_D4 = 1;
  LCD_DIR_D5 = 1;
  LCD_DIR_D6 = 1;
  LCD_DIR_D7 = 1;
  LCD_D4 = data_terima.bit.b0;
  LCD_D5 = data_terima.bit.b1;
  LCD_D6 = data_terima.bit.b2;
  LCD_D7 = data_terima.bit.b3;
  LCD_E = 0;
  LCD_E = 1;
  lcd_tunggu();
  LCD_E = 0;
  LCD_DIR_D4 = 0;
  LCD_DIR_D5 = 0;
  LCD_DIR_D6 = 0;
  LCD_DIR_D7 = 0;
}
void lcd_kirim_perintah(char rs, char data)
{
  union byte_def data_terima;
  data_terima.byte = data;
  LCD_RS = rs;
  LCD_DIR_D4 = 1;
  LCD_DIR_D5 = 1;
  LCD_DIR_D6 = 1;
  LCD_DIR_D7 = 1;
  LCD_D4 = data_terima.bit.b4;
  LCD_D5 = data_terima.bit.b5;
  LCD_D6 = data_terima.bit.b6;
  LCD_D7 = data_terima.bit.b7;
  LCD_E = 0;
  LCD_E = 1;
  lcd_tunggu();
  LCD_E = 0;
  lcd_tunggu();
  LCD_D4 = data_terima.bit.b0;
  LCD_D5 = data_terima.bit.b1;
  LCD_D6 = data_terima.bit.b2;
  LCD_D7 = data_terima.bit.b3;
  LCD_E = 0;
  LCD_E = 1;
  lcd_tunggu();
  LCD_E = 0;
  lcd_tunggu();
  LCD_DIR_D4 = 0;
  LCD_DIR_D5 = 0;
  LCD_DIR_D6 = 0;
  LCD_DIR_D7 = 0;
}
```

```
void lcd_kirim_karakter(char data)
{
  lcd_kirim_perintah(0, 0b00000110); //mode entri
  lcd_kirim_perintah(1, data); //tulis data
}
void lcd_kirim(char *data)
{
  char k = 0, kode;

  lcd_kirim_perintah(0, 0b00000110); //mode entri

  kode = data[k]; //baca data
  while(kode != 0)
  {
   lcd_kirim_perintah(1, kode); //tulis data ke LCD
   k++;
   kode = data[k]; //baca data berikutnya
  }
}
void lcd_hapus_layar(void)
{
  lcd_kirim_perintah(0, 0b00000001); //clear
}

void lcd_posisi_awal(void)
{
  lcd_kirim_perintah(0, 0b00000010); //home
}

void lcd_posisi_baris_1(char x)
{
  lcd_kirim_perintah(0, 0b10000000 + x); //home
}

void lcd_posisi_baris_2(char x)
{
  lcd_kirim_perintah(0, 0b11000000 + x); //home
}

void lcd_hidup_tanpa_kursor(void)
{
  lcd_kirim_perintah(0, 0b00001100);
}

void lcd_blank(void)
{
  lcd_kirim_perintah(0, 0b00001000);
}
```

```
void lcd_mati(void)
{
  LCD_ON = 1;
}

void lcd_hidup(void)
{
  LCD_ON = 0;
}

void lcd_inisialisasi(void)
{
  //inisialisasi port:
  LCD_DIR_ON = 1;
  LCD_ON = 0; //aktif rendah
  LCD_DIR_D4 = 0;
  LCD_DIR_D5 = 0;
  LCD_DIR_D6 = 0;
  LCD_DIR_D7 = 0;
  LCD_DIR_E  = 1;
  LCD_E = 0;

  LCD_DIR_RS = 1;
  //inisialisasi mode 4 bit:
  lcd_tunggu_lama();
  lcd_kirim_perintah_8_bit(0, 0b0011);
  lcd_tunggu_lama();
  lcd_kirim_perintah_8_bit(0, 0b0011);
  lcd_tunggu_lama();
  lcd_kirim_perintah_8_bit(0, 0b0011);
  lcd_kirim_perintah_8_bit(0, 0b0010);
  lcd_kirim_perintah(0, 0b00101000); //4 bit
  lcd_kirim_perintah(0, 0b00001000); //off
  lcd_kirim_perintah(0, 0b00000001); //clear
  lcd_kirim_perintah(0, 0b00000110); //entri
}

void lcd_kirim_bulat(int x)
{
  char bil1, bil2, bil3, bil4;
  //jika diperlukan, beri tanda negatif:
  if(x < 0)
  {
    lcd_kirim_karakter('-');
    x = -x;
  }
  //hitung masing-masing digit:

  bil3 = x/100; // ratusan
  x = x - 100*bil3;
  bil4 = x/10; // puluhan
  x = x - 10*bil4; // satuan

  //kirim masing-masing digit:

    lcd_kirim_karakter(bil3 + 48);

   lcd_kirim_karakter(bil4 + 48);
                lcd_kirim_karakter(x + 48);
  if(ti_u1c1 == 1) u1tb = 0;
}

void lcd_kirim_hex(unsigned int x)
{
  char bil1;
  //jika diperlukan, beri tanda negatif:
  if(x < 0)
  {
    lcd_kirim_karakter('-');
    x = -x;
  }
   //hitung masing-masing digit:

  bil1 = x/16; // puluhan
   x = x - 16*bil1; // satuan
   //kirim masing-masing digit:
 lcd_kirim_karakter(bil1 + 48);

  lcd_kirim_karakter(x + 48);
  if(ti_u1c1 == 1) u1tb = 0;
 }
 void lcd_kirim_hex_3(unsigned int x)
 {
  char bil1,bil2;
  //jika diperlukan, beri tanda negatif:
  if(x < 0)
  {
    lcd_kirim_karakter('-');
    x = -x;
  }
   //hitung masing-masing digit:

 bil1=x/256;
                x=x-256*bil1;
  bil2 = x/16; // puluhan
  x = x - 16*bil2; // satuan
   //kirim masing-masing digit:
if((bil1>=0)&&(bil1<=9))
                lcd_kirim_karakter(bil1 + 48);
else
                lcd_kirim_karakter(bil1 + 55);
if((bil2>=0)&&(bil2<=9))
                lcd_kirim_karakter(bil2 + 48);
else
```

```
        lcd_kirim_karakter(bil2 + 55);
if((x>=0)&&(x<=9))
        lcd_kirim_karakter(x + 48);
else
        lcd_kirim_karakter(x + 55);
}
```

SUBRUTIN RTC

```
#include    "sfr_r813.h"                /* Definition of the R8C/13 SFR */
#include    "lic_Bus.h"
#include "seting_port.h"
//;+++++++++++++++++++++++++++++++++++++++++++++++++
//; definisi alamat RTC SAMA DENGAN YANG DI SETJAM
//;+++++++++++++++++++++++++++++++++++++++++++++++++

#define RTCsecond                   0x00;
#define RTCminute                   0x01;
#define RTChour                             0x02;
#define RTCday_of_week              0x03;
#define RTCdate                             0x04;
#define RTCmonth                    0x05;
#define RTCyear                             0x06;
#define RTCCtrlReg                  0x07;


extern unsigned char second;

extern unsigned char minute;
unsigned char hour;
unsigned char day_of_week;
unsigned char date;
unsigned char month;
unsigned char year;
unsigned char ctrl;

static unsigned char WriteData[8];
static unsigned char ReadData[8];
licPack licData_w;
licPack licData_r;

void SetJam(void)
{
        unsigned char temp;

        initlicBus();
        WriteData[0] = 0x01;        /* Detik */
        WriteData[1] = 0x37;        /* Menit */
        WriteData[2] = 0x07;        /* Jam */
        WriteData[3] = 0x02;        /* Hari */
        WriteData[4] = 0x10;        /* Tanggal */
        WriteData[5] = 0x11;        /* Bulan */
        WriteData[6] = 0x09;                                    /* Tahun */
        WriteData[7] = 0x10;                                    /* Kontrol Kedip */
        licData_w.iic_DeviceAddress = 0xD0;     //alamat RTC
        licData_w.iic_MemoryAddress = RTCsecond; //awal alamat yg dibaca/tulis
        licData_w.iic_Data = WriteData;         //tulis writedata[] ke alamat yg dituju!!
        licData_w.iic_NumberOfByte = 8;                         //jml byte data
        //  if (licBusWrite(&licData_w) == ACK);
        temp = licBusWrite(&licData_w);

}

void ReadTime (void)
{
        licData_r.iic_DeviceAddress = 0xD0;                     //alamat rtc
        licData_r.iic_MemoryAddress = RTCsecond;   //awal alamat yg dibaca

        licData_r.iic_Data =ReadData;
        licData_r.iic_NumberOfByte = 8;                         //jml byte data
        if(licBusRead(&licData_r) == ACK)
        //  licBusRead(&licData_r);
        {
            second = ReadData[0];
            minute = ReadData[1];
            hour = ReadData[2];     //readdata=[second;minute;hour]
            day_of_week = ReadData[3];
            date = ReadData[4];
            month = ReadData[5];
            year = ReadData[6];
            ctrl = ReadData[7];
        }
}
```

SUBRUTIN KOMUNIKASI I2C

```
/*****************************************************************************
*       File Name   : iic_bus.c                                              *
*       Contents    : IIC Bus file                                           *
*       Copyright   : RENESAS TECHNOLOGY CORPORATION                         *
*                                       AND RENESAS SOLUTIONS CORPORATION    *
*       Version             : 1.0                                            *
*       note                :                                                *
*****************************************************************************/

#include    "sfr_r813.h"                                /* Definition of the R8C/13 SFR */
#include    "lic_Bus.h"
                                        //#define iic_sda_d     pd1_7
                                        //#define iic_sda               p1_7
#define iic_sda_d       pd1_1
```

```
#define iic_sda                 p1_1
//#define iic_scl_d     pd1_6
//#define iic_scl               p1_6

#define iic_scl_d       pd1_0
#define iic_scl         p1_0

void _WaitTime0us(void);
void _WaitTime1us(void);
void _WaitTime2us(void);

#define _Wait_tHIGH             _WaitTime1us()      /* Clock pulse width high */
#define _Wait_tLOW              _WaitTime2us()      /* Clock pulse width low */
#define _Wait_tHD_STA           _WaitTime1us()      /* Start hold time */
#define _Wait_tSU_STA           _WaitTime1us()      /* Start setup time */
#define _Wait_tHD_DAT           _WaitTime0us()      /* Data in hold time */
#define _Wait_tSU_DAT           _WaitTime1us()      /* Data in setup time */
#define _Wait_tAA               _WaitTime1us()      /* Access time */
#define _Wait_tSU_STO           _WaitTime1us()      /* Stop setup time */
#define _Wait_tBUF              _WaitTime2us()      /* Bus free time for next mode */
void initlicBus(void)
{
        iic_sda_d = 0;                  /* SDA input ("H" state) */
        iic_scl_d = 0;                  /* SCL input ("H" state) */
}
void StartCondition(void)
{
        iic_scl_d = 1;                  /* SCL output */
        iic_scl = 0;                    /* SCL="L" */
        _WaitTime1us();                 /* wait *1 */
        iic_sda_d = 0;                  /* SDA="H" */
        _WaitTime1us();                 /* wait */
        _WaitTime1us();                 /* wait *! */
        iic_scl = 1;                    /* SCL="H" */
        _Wait_tSU_STA;                  /* wait */
        iic_sda_d = 1;                  /* SDA output */
        iic_sda = 0;                    /* SDA="L" */
        _Wait_tHD_STA;                  /* wait */
        _WaitTime1us();                 /* wait *1 */
        iic_scl = 0;                    /* SCL="L" */
}

void StopCondition(void)
{       iic_scl_d = 1;                  /* SCL output */
        iic_scl = 0;                    /* SCL="L" */
        _WaitTime1us();                 /* wait *1 */
        iic_sda_d = 1;                  /* SDA output */
        iic_sda = 0;                    /* SDA="L" */
        _WaitTime1us();                 /* wait *1 */
        iic_scl = 1;                    /* SCL="H" */
        _Wait_tSU_STO;                  /* wait */
        iic_sda_d = 0;                  /* SDA="H" */
        _WaitTime1us();                 /* wait */
        _WaitTime1us();                 /* wait *1 */
        iic_scl = 0;                    /* SCL="L" */
}
unsigned char licBusRead(licPack *licData)
{
        unsigned char i,ret;

        /* Random Read Cycle / Sequential Ramdom Read Cycle */
        licData->iic_DeviceAddress &= 0xFE;
        /* WRITE Setting Device Address */
        StartCondition();               /* Start Condition */
        while (1)
        {
        if ((ret=ByteWrite(licData->iic_DeviceAddress)) == NOACK)/* WRITE Device Address */
        break;                          /* NoAck Detect */
if ((ret=ByteWrite(licData->iic_MemoryAddress)) == NOACK)       /* WRITE Memory Address */
        break;                          /* NoAck Detect */
        licData->iic_DeviceAddress |= 0x01;
                /* READ Setting Device Address */
        StartCondition();               /* ReStart Condition */
        if ((ret=ByteWrite(licData->iic_DeviceAddress)) == NOACK)/* WRITE Device Address */
        break;                          /* NoAck Detect */
        for (i=0; i<licData->iic_NumberOfByte; i++)             /* specified bytes as loop */
        {
                ByteRead(licData->iic_Data, ACK);
        /* Read data (Ack output) */
                licData->iic_Data++;
        }
        ByteRead(licData->iic_Data, NOACK);
        /* Read data (NoAck output) */
        break;
        }
        StopCondition();                /* Stop Condition */
        return(ret);
}
unsigned char licBusWrite(licPack *licData)
{
        unsigned char i,ret;

        /* Byte Write / Page Write */
        licData->iic_DeviceAddress &= 0xFE;
        /* WRITE Setting Device Address */
        StartCondition();               /* Start Condition */
        while (1)
        {
```

```
        if ((ret=ByteWrite(licData->iic_DeviceAddress)) == NOACK)/* Write Device Address */
        break;
                        /* NoAck Detect */
if ((ret=ByteWrite(licData->iic_MemoryAddress)) == NOACK)       /* Write Memory Addreess */
        break;
                        /* NoAck Detect */
        for (i=0; i<licData->iic_NumberOfByte; i++)             /* specified bytes as loop */
        {
        if ((ret=ByteWrite(*(licData->iic_Data))) == NOACK)     /* Write Data */
        break;
                /* NoAck Detect */
        licData->iic_Data++;
        }
        break;
        }
        StopCondition();
                        /* Stop Condition */
        return(ret);
}
unsigned char ByteWrite(unsigned char iic_writeData)
{
        unsigned char maskData=0x80;            /* MSB first */
        unsigned char ret=ACK;                          /* Ack/NoAck */

        while (maskData)
        {
                iic_sda = 0;                            /* initialize port-latch */
                if (iic_writeData & maskData)           /* "H" output ? */
                {
                iic_sda_d = 0;
                }
                else
                {
                        iic_sda_d = 1;
        /* No  SDA="L" */
                        asm("nop");
        /* wait *1 */
                        asm("nop");
        /* wait *1 */
                        asm("nop");
        /* wait *1 */
                        asm("nop");
        /* wait *1 */
                }
                _Wait_tSU_DAT;
        /* wait */
                _WaitTime1us();
        /* wait *1 */
                iic_scl = 1;                            /* SCL="H" */
                _Wait_tHIGH;
        /* wait */
                _WaitTime1us();
        /* wait *1 */
                iic_scl = 0;                            /* SCL="L" */
                maskData >>= 1;
        /* change mask data */
                _WaitTime1us();                                 /* wait *1 */
        }

        iic_sda_d = 0;
        /* SDA input */
        _Wait_tAA;
        /* wait */
        _WaitTime2us();
        /* wait *1 */
        iic_scl = 1;                                    /* SCL="H" */
        if (iic_sda)
         ret=NOACK;                                     /* NoAck Detect */
        _Wait_tHIGH;
        /* wait */
        _WaitTime1us();
        /* wait *1 */
        iic_scl = 0;                                    /* SCL="L" */
        _Wait_tHD_DAT;
        /* wait */
        return(ret);
}
void ByteRead(unsigned char *iic_readData, unsigned char ackData)
{
        unsigned char maskData=0x80;                    /* MSB first */
        unsigned char readData;
        //unsigned int a;

        *iic_readData = 0;
        while (maskData)
        {
                readData = *iic_readData | maskData;    /*  */
                iic_sda_d = 0;
        /* initialize port-latch */
                _Wait_tAA;
        /* wait */
                iic_scl = 1;                            /* SCL="H" */
        while(iic_scl_d==0);//wait clock strectching;
                if (iic_sda)                            /* SDA="H" ? */
                {
                        *iic_readData = readData;       /* Yes  */
                }
                else
                {
                        asm("nop");
        /* wait *1 */
```

```
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
                asm("nop");
/* wait *1 */
            }
            _Wait_tHIGH;
/* wait */
            _WaitTime1us();
/* wait *1 */
            iic_scl = 0;
*/
            maskData >>= 1;
/* Change mask data */
            _WaitTime1us();
/* wait *1 */
//          a=0xff;
//          while(a>0)a--;

}
if (!ackData)
/* Ack output ? */
{
/* Ack output */
            iic_sda = ACK;
/* Yes SDA="L" */
                    iic_sda_d = 1;
/* SDA output */
}
else
{
/* NoAck output */
                    iic_sda = NOACK;                    /* No  SDA="H" */
                    iic_sda_d = 0;
/* SDA input */
}
_Wait_tSU_DAT;
/* wait */
_WaitTime1us();
/* wait *1 */
iic_scl = 1;                                            /* SCL="H" */
_Wait_tHIGH;
/* wait */
_WaitTime1us();
/* wait *1 */
iic_scl = 0;                                            /* SCL="L" */
iic_sda_d = 0;
/* SDA input */
_WaitTime1us();
/* wait *1 */
}
void _WaitTime0us(void)
{
}
void _WaitTime1us(void)
{
/* +14cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle = 20cycle */
}
void _WaitTime2us(void)
{
/* +14cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
asm("nop");                 /* +1cycle */
```

```
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle */
        asm("nop");             /* +1cycle = 40cycle */
}
```

SUBRUTIN SDCARD

```
#include    "sfr_r813.h"                /* Definition of the R8C/13 SFR */
#include    "command.h"
#include    "mmc.h"
#include    "FAT.h"

unsigned char sector[512]={};
unsigned char response = 0xff;
unsigned char response2 = 0xff;

unsigned long sector_partisi_pertama;

extern unsigned int reserved_sector;    // dari FAT.c
void init_SPI()
{       DIN         = HIGH;
        CLK         = HIGH;
        CS          = HIGH;

        d_DIN       = OUTPUT;
        d_DOUT      = INPUT;
        d_CLK       = OUTPUT;
        d_CS        = OUTPUT;

        u0mr        = 0x01;
        u0c0        = 0x8a;             // f32SIO
        u0brg       = 1;                // 20 MHz/32/(2*(1+1)) = 156250 Hz

        u0irs       = 1;                // interrupt saat transmission completed
}

void uart_trans(unsigned char c)
{       while( !ti_u0c1 );
        u0tb = c;
        while( !ir_s0tic );
        ir_s0tic = 0;
}

unsigned char uart_rec()
{       uart_trans(0xff);               // DIN dijaga HIGH terus selama receive
        //while( !ri_u0c1 );
        while ( !ir_s0ric );
        ir_s0ric = 0;
        return( (unsigned char) u0rb);
}

void enable_rec()
{       while( !ti_u0c1 );
        while( !txept_u0c0 );
        re_u0c1 = 1;
}

void enable_trans()
{
        te_u0c1 = 1;
}

void uart_end()
{       while( !ti_u0c1 );
        while( !txept_u0c0 );
        re_u0c1 = 0;
        te_u0c1 = 0;
}

void send_cmd(unsigned char cmd, unsigned long addr, unsigned char crc)
{
        unsigned char frame[6];
        unsigned char temp;
        signed char x=5;

        frame[0] = cmd;

        for(x=3; x>=0; x--)
        {       temp = (char) (addr>>(8*x));
                frame[4-x] = temp;
        }

        frame[5] = crc;

        for(x=0; x<6; x++) uart_trans(frame[x]);

}


char init_MMC()
{       unsigned char i=0;
        char response = 0x02;
```

```
                                        // DI, CS sdh HIGH sebelumnya
        enable_trans();
        for(i=0; i<11; i++) uart_trans(0xff);           // beri minimal 74 pulsa

        CS          = LOW;
                    // MMC sampling jika CS LOW --> SPI


        send_cmd(MMC_GO_IDLE_STATE,0x00000000,0x95);

        i=0;
        enable_rec();
                    // siap menerima data

        response = get_response();
        if (response != 0x01) return MMC_INIT_ERROR;        // resp harus 0x01 : idle state

                    while(response==0x01)           // terus kirim CMD1 sampai resp 0x00
        {
        CS          = HIGH;
        uart_trans(0xff);
        CS  = LOW;

                    send_cmd(MMC_SEND_OP_COND,0,0xff);
                    response = get_response();
}

        CS          = HIGH;
        uart_trans(0xff);
        uart_end();
        return MMC_SUCCESS;                                 // 0x00
}

char get_response()
{       char i=0;
        char response;

        while(i<=8)                                         // dibatasi
selama NCR
        {       response = uart_rec();
                if(response == 0x00) break;
                if(response == 0x01) break;
                i++;
        }
        return response;
}


char mmcReadRegister(unsigned char cmd)
{       unsigned char jumlah=0;
        unsigned int i=0;
```

```
                                        char rvalue = MMC_TIMEOUT_ERROR;

        CS          = LOW;
        enable_trans();


        send_cmd(cmd,0,0xff);

        enable_rec();

        i=0;

        if(get_response() == 0x00)
        {       i=0;

                if(get_xx_response(0xfe) == 0xfe)
        /*      while(i<=1000)
                {       response2 = uart_rec();
                        if(response2 == 0xfe) break;                    //tunggu response 0xfe
                        i++;
                }
        */
                for (jumlah=0; jumlah < 16; jumlah++) sector[jumlah] = uart_rec();
                uart_trans(0xff);                       // tangkap CRC jika for di atas cuma 14 kali
                uart_trans(0xff);                       //
                rvalue = MMC_SUCCESS;
        }
        else rvalue = MMC_RESPONSE_ERROR;

        CS = HIGH;
        uart_trans(0xff);
        uart_end();

        return rvalue;

}

char get_xx_response(char resp)
{       int i=0;
        char response;

        while(i<=1000)
        {       response = uart_rec();
                if(response==resp) break;
                i++;
        }
        return response;
}

char read_block(unsigned long addr)                     // addr adalah alamat dlm sector, bukan byte
```

```
{       unsigned int i=0;                                       // u/ cari sector secara
logic
        char rvalue = MMC_RESPONSE_ERROR;     // tdk u/ sector scr fisik

        // read_block(0); --> baca VBR

        addr = (sector_partisi_pertama + addr)*512UL;

        do{
        CS = LOW;
        enable_trans();
        enable_rec();

        send_cmd(MMC_READ_SINGLE_BLOCK, addr, 0xff);
        if(get_response() == 0x00)                      // R1 0x00 jika tidak ada error
        {
                // tunggu start data block token
                if(get_xx_response(MMC_START_DATA_BLOCK_TOKEN) ==
MMC_START_DATA_BLOCK_TOKEN)
                {

                        for(i=0; i<512; i++) sector[i] = uart_rec();          // data sesungguhnya

                        uart_rec();                 // 2 byte CRC
                        uart_rec();
                        rvalue = MMC_SUCCESS;
                }
                else rvalue = MMC_RESPONSE_ERROR;     // start token tidak prnh diterima
        }
        else rvalue = MMC_RESPONSE_ERROR;     // R1 error

        CS = HIGH;
        uart_rec();
        uart_end();
        }while(rvalue != MMC_SUCCESS);
        return rvalue;
}

char read_block_fisik(unsigned long addr)               // addr adalah alamat dlm sector, bukan byte
{       unsigned int i=0;                                       // u/ cari sector secara fisik
        char rvalue = MMC_RESPONSE_ERROR;     // tdk u/ sector scr logic

        // read_block(0); --> baca VBR
        addr = addr*512UL;

        do{
        CS = LOW;
        enable_trans();
        enable_rec();

        send_cmd(MMC_READ_SINGLE_BLOCK, addr, 0xff);
        if(get_response() == 0x00)                      // R1 0x00 jika tidak ada error
        {
                // tunggu start data block token
                if(get_xx_response(MMC_START_DATA_BLOCK_TOKEN) ==
MMC_START_DATA_BLOCK_TOKEN)
                {

                        for(i=0; i<512; i++) sector[i] = uart_rec();          // data sesungguhnya

                        uart_rec();                 // 2 byte CRC
                        uart_rec();
                        rvalue = MMC_SUCCESS;
                }
                else rvalue = MMC_RESPONSE_ERROR;     // start token tidak prnh diterima
        }
        else rvalue = MMC_RESPONSE_ERROR;     // R1 error

        CS = HIGH;
        uart_rec();
        uart_end();
        }while(rvalue != MMC_SUCCESS);
        return rvalue;
}

char cek_busy()                                         //   |---------------|-------|
{       char i=0;                                       //   |data_response  | busy  |
        char response;                                  //   |_______________|_______|
        char rvalue;

        enable_trans();
        enable_rec();

        while(i<=8)
        {       // tunggu data_response
                response = uart_rec();
                response &= 0x1f;
                switch(response)
                { case 0x05: rvalue = MMC_SUCCESS;break;
                  case 0x0b: return (MMC_CRC_ERROR);
                  case 0x0d: return (MMC_WRITE_ERROR);

                  // belum dikasikan nilai default
                }
                if(rvalue == MMC_SUCCESS) break;
                i++;
        }
```

```
        do
                // tunggu busy
        { response = uart_rec();                         // 0x00 = busy

                // tidak ada batasan waktu
        }while(response == 0x00);               // klo perlu, kasi batas waktu
        return response;
}

char data_resp()
{       char i=0;                               //      |data_response        | busy  |
        char response;                          //      |_____________________|_______|
        char rvalue;

        enable_trans();
        enable_rec();

        while(i<=8)
        // tunggu data_response
        {       response = uart_rec();
                response &= 0x1f;
                switch(response)
                { case 0x05: rvalue = MMC_SUCCESS;break;
                  case 0x0b: return (MMC_CRC_ERROR);
                  case 0x0d: return (MMC_WRITE_ERROR);

                  // belum dikasikan nilai default
                }
                if(rvalue == MMC_SUCCESS) break;
                i++;
        }

// jika ada error saat write_multiple, lebih baik stop transmisi dengan CMD12
// ref; halaman 5-15
}

//addr adalah nomer sector logic
char write_block(unsigned long addr)            // addr dlm sector, bkn byte
{       unsigned int i=0;
        char rvalue = MMC_RESPONSE_ERROR;

        addr = (sector_partisi_pertama + addr)*512UL;           // konversi ke byte
(logic)

        CS = LOW;
        enable_trans();
        send_cmd(MMC_WRITE_BLOCK, addr, 0xff);

        enable_rec();
        if(get_response() == 0x00)              // R1 0x00 jika tidak ada error
        {       uart_trans(0xff);               // tunggu NWR
                uart_trans(0xfe);               // start block token

                for(i=0; i<512; i++) uart_trans( sector[i] );           // kirim 512 byte

                uart_rec();             // 2 byte CRC
                uart_rec();

                cek_busy();
                rvalue = MMC_SUCCESS;
        }
        else rvalue = MMC_RESPONSE_ERROR;                       // R1 error

        CS = HIGH;
        uart_rec();                             // kasi 8 clock, untuk MMC selesaikn proses internal
        uart_end();
        return rvalue;
        }

char write_block_fisik(unsigned long addr)              // addr dlm sector, bkn byte
{       unsigned int i=0;
        char rvalue = MMC_RESPONSE_ERROR;

        addr = addr*512UL;              // konversi ke byte (logic)

        CS = LOW;
        enable_trans();
        send_cmd(MMC_WRITE_BLOCK, addr, 0xff);

        enable_rec();
        if(get_response() == 0x00)              // R1 0x00 jika tidak ada error
        {       uart_trans(0xff);               // tunggu NWR
                uart_trans(0xfe);               // start block token

                for(i=0; i<512; i++) uart_trans( sector[i] );           // kirim 512 byte

                uart_rec();             // 2 byte CRC
                uart_rec();

                cek_busy();
                rvalue = MMC_SUCCESS;
        }
        else rvalue = MMC_RESPONSE_ERROR;               // R1 error

        CS = HIGH;
        uart_rec();                             // kasi 8 clock, untuk MMC selesaikn proses internal
        uart_end();
        return rvalue;
}

char offset_partisi(void)               // mencari nilai variabel sector_pertisi_pertama
{                                       // baca sector 0 fisik
        unsigned int i = 0;
```

```
        unsigned long offset = 0;
        unsigned char charBuf;

        CS = LOW;
        enable_trans();
        send_cmd(MMC_READ_SINGLE_BLOCK, 0, 0xff);

        enable_rec();
        if(get_response() == 0x00)                  // R1 0x00 jika tidak ada error
        {
                // tunggu start data block token
                if(get_xx_response(MMC_START_DATA_BLOCK_TOKEN) ==
MMC_START_DATA_BLOCK_TOKEN)
                {
                        for(i=0; i<454; i++) uart_rec();        // tidak di ambil;

                        charBuf = uart_rec();
                // 454
                        offset = charBuf;
                        charBuf = uart_rec();
                // 455
                        offset |= (unsigned long int)charBuf << 8;
                        charBuf = uart_rec();
                // 456
                        offset |= (unsigned long int)charBuf << 16;
                        charBuf = uart_rec();
                // 457
                        offset += (unsigned long int)charBuf << 24;

                        for(i = 458; i < 512; i++)              uart_rec();

                        uart_rec();             // 2 byte CRC
                        uart_rec();
                }
        }

        sector_partisi_pertama = offset;
        CS = HIGH;
        uart_rec();                     // kasi 8 clock, untuk MMC selesaikn proses internal
        uart_end();
        return (char)offset;

}
```

SUBRUTIN FAT

```
#include    "FAT.h"
#include    "command.h"
#include    "sfr_r813.h"
#include    "mmc.h"


extern unsigned char sector[512];

extern unsigned int adc1;
extern unsigned int adc2;

unsigned int desimal;
unsigned char minute;
extern unsigned char second;
extern unsigned char hour;
extern unsigned char year;
extern unsigned char month;
extern unsigned char date;
extern unsigned char response;
unsigned int clust_x;
unsigned int jmlh_clust;

unsigned int jmlh_clust_kosong;

// dipake o/ cari_cluster_kosong()
unsigned int returnCluster;

#define BUFFER_SIZE 2
#define SECTOR_SIZE 512

// seluruhnya 44 byte, dalam format LITTLE-ENDIAN


extern unsigned long sector_partisi_pertama;
/**
 *          the sector in the MMC/SD card where data starts.
 *          This is right after the root directory sectors in FAT16
 **/
unsigned long data_start;
/**
 *          the first sector where entries in the root directory is
 *          stored in FAT16.
 **/
 unsigned long root_directory;
/**
 *          the number of sectors per File Allocation Table (FAT)
 **/
 unsigned long sector_per_FAT;
/**
 *          the number of sectors per cluster for the MMC/SD card
 **/
 unsigned int sector_per_cluster;
/**
 *          the number of reserved sectors in the MMC/SD card
 **/
 unsigned int reserved_sector;
/**
 *          the number of File Allocation Table (FAT) on the MMC/SD.
 *          this is usually 2
 **/
```

```
unsigned int num_of_FAT;
/**
 *		the number of sectors in the section for root directory
 *		entries.
 **/
unsigned int root_sector;
/**
 *		FAT sector number last searched for an empty cluster
 **/
unsigned long int last_FAT_sector_searched_for_empty;
/**
 *		initialize variables for file system
 *
 *		@param	buf			the buffer to be used to access the MMC/SD card
 *
 *		@return	0			variables sucessfully initialized
 *		@return	-1			unknown file system
 *		@return -2			error reading from MMC/SD card
 **/

char read_file_system_info()
{
		unsigned int max_root_entry = 0;
		unsigned long int tot_sec_16;


		read_block(0);
		// Baca VBR

		sector_per_cluster = sector[13];
		reserved_sector = sector[14];
		reserved_sector |= (unsigned long int)sector[15] << 8;
		num_of_FAT = sector[16];

		max_root_entry = sector[17];
		max_root_entry |= (unsigned int)sector[18] << 8;

		tot_sec_16 = sector[19];
		// cari jumlah sector, dari VBR smpi volume habis
		tot_sec_16 |= (unsigned int)sector[20] << 8;
		if(!tot_sec_16)
		{
		// jika di TotSec32
			tot_sec_16 = sector[32];
			tot_sec_16 |= (unsigned long int)sector[33] << 8;
			tot_sec_16 |= (unsigned long int)sector[34] << 16;
			tot_sec_16 |= (unsigned long int)sector[35] << 24;
		}

		sector_per_FAT = sector[22];
		sector_per_FAT |= (unsigned int)sector[23] << 8;
		root_directory = reserved_sector + sector_per_FAT * num_of_FAT;	// # sect root dari VBR
		root_sector = ( max_root_entry / SECTOR_SIZE ) * 32;			// jumlh
		//sector root
		data_start = root_directory + root_sector;
		jmlh_clust = (tot_sec_16 - data_start)/sector_per_cluster;
		return 0;
}

// cari jumlah clust yg masih kosong
// variabel jmlh_clust_kosong adalah variabel global
unsigned int hit_jum_clust_kosong()
{ unsigned int j;
  unsigned long i;

   jmlh_clust_kosong=0;
	for(i=reserved_sector; i<(sector_per_FAT + reserved_sector); i++)
	{		read_block(i);

			if(i != (sector_per_FAT + reserved_sector - 1))
			{
			  for(j=0; j<512; j += 2)
			  {
				if( sector[j] == 0x00 && sector[j+1] == 0x00 )
				jmlh_clust_kosong++;
			  }
			}else
			{
			  for(j=0; j< ((256+2 - ((256*sector_per_FAT) - (jmlh_clust)))*2); j+=2)
			  {
				if( sector[j] == 0x00 && sector[j+1] == 0x00 )
				jmlh_clust_kosong++;
			  }
			}
	}
	return jmlh_clust_kosong;
//	jmlh_clust_kosong -= (256*sector_per_FAT) - jmlh_clust;
}

// Konversi cluster jadi sector (terhitung dari sector 0 logic)
unsigned long clust2sect(unsigned int clust)
{		clust -=2;
		return (unsigned long) clust*sector_per_cluster + data_start;
}


// jika ada entry yang kosong, return entry tsb
// jika tdk ada yg kosong,  return 1000
unsigned int cari_entry_kosong()
{		char j,k;
		unsigned int nomer_entry = 0;			// 0 s/d 511
```

```
            for(k=0; k<32; k++)                                     // ulang 32
sector root_dir
            {           read_block(root_directory+k);               // dari sector pertama s.d. trakhir
root_dir

                        for(j=0; j<16; j++)
                        {           if( sector[j*32]==0x00 || sector[j*32]==0xe5) return nomer_entry;
                                    else nomer_entry++;
                        }
            }
            return 1000;
}

// return nomer cluster yg kosong
// jika tidak ada yg kosong, return 0
// setiap kali fungsi ini dipanggil, maka returnCluster selalu update
// waspada jika mau panggil fungsi. pastikan, benar2 ingin mancari cluster kosong
unsigned int cari_cluster_kosong()
{           unsigned long sect, end_sect;               // # sector dari FAT, jangan ambigu dengan
sector[]

            unsigned int temp;
            unsigned int i;
            unsigned char j;

   if (last_FAT_sector_searched_for_empty == 0)
   {
      sect = reserved_sector;
   }
   else
   {
      sect = last_FAT_sector_searched_for_empty;
   }
   end_sect = reserved_sector + sector_per_FAT;

   // 256 entries in a 512-byte FAT sector
  // returnCluster = (sect - reserved_sector) * 256;


            temp = (sect - reserved_sector) * 256;
            if(returnCluster != 0)
            {
                        i = ((returnCluster-temp)+1)*2;
                        returnCluster++;
            }
            else i = 0;

   while (1)
   {
      while(sect < end_sect)
      {
         while(read_block(sect) != MMC_SUCCESS);        // baca sampai benar2 dpt

for(i ; i < SECTOR_SIZE; i += 2)
{
   if(sector[i] == 0x00 && sector[i+1] == 0x00)
   {
      // Remember where we found it:
      last_FAT_sector_searched_for_empty = sect;
                                                                  if( (returnCluster+1)%256 == 0 )
                                                                  { sect++;
                                                                    last_FAT_sector_searched_for_empty++;
                                                                  }
      return returnCluster;
   }
   returnCluster++;
}
sect++;                                                 i=0;
   }

   // We reached the end of the FAT.  Should we wrap around and search from
   // the beginning?
   if (last_FAT_sector_searched_for_empty > reserved_sector)
   {
      // Yes.
      sect = reserved_sector;
      end_sect = last_FAT_sector_searched_for_empty;
      last_FAT_sector_searched_for_empty = 0;
      returnCluster = 0;
                                                        temp = 1;
      // We will take the else block next time if we don't find anything.
   }
   else
   {
      return 0;
   }
  }
}

//coba tulis file di cluster data,cuma untuk mengosongkan cluster
unsigned int tulis_data_awal()
{           int j;
   char j;

unsigned long sect_clust_x;

      clust_x = cari_cluster_kosong();
      sect_clust_x = clust2sect(clust_x);
      read_block(sect_clust_x);

      sector[0]=0x0A;                                                   //penanda data awal
      for(i=1;i<512;i++)
```

```
    {sector[i]=0x00;
            }
             sector[511]=0x0A;
            //tulis data 00
            write_block(sect_clust_x);
            //karena maw kosongkan 1 cluster, maka diulang 32 kali.
            //mulai sect_clust_x sampai sect_clust_x+31
            //jadi kita for mulai j=1 sampe j=31

            for(j=1;j<=31;j++)
            {
                    read_block(sect_clust_x+j);

                    sector[0]=0x0A;
                    for(i=1;i<511;i++)
             {sector[i]=0x00;}
                     sector[511]=0x0A;

                    write_block(sect_clust_x+j);
            }

}
unsigned int tulis_data()//ini adalah fungsi tulis data sebenarnya
{           int i,j,hasilkonversi;
            char bil1,bil2,bil3,bil4,bil5,bil6;   //bwt hasil adc
            char jam1,jam2,mnt1,mnt2;
            unsigned long sect_clust_x,sect_clust_skrg;

                                /*ini diedit untuk mengakses sector baru dalam 1 cluster*/
            sect_clust_x = clust2sect(clust_x);
            sect_clust_skrg = sect_clust_x;
            for(i=1;i<=32;i++)
            {           read_block(sect_clust_skrg);
                        if (sector[501]==0x00)
                        {//sect_clust_skrg=sect_clust_x;
                        i=33;
                        break;}

                        else //berati sector tsb dah penuh.,ganti sector slanjutnya
                        sect_clust_skrg=sect_clust_skrg+1;
            }
                                                /*selesai*/
            read_block(sect_clust_skrg);
            hasilkonversi=adc1;
            //konversi adc 0-3FF ke bentuk satuan
            bil1=hasilkonversi/256;
            hasilkonversi=hasilkonversi-256*bil1;
            bil2=hasilkonversi/16;
            bil3=hasilkonversi-16*bil2;

            if((bil1>=0)&&(bil1<=9))
                        bil1=bil1 + 48;
            else
                        bil1=bil1 + 55;
            if((bil2>=0)&&(bil2<=9))
                        bil2=bil2 + 48;
            else
                        bil2=bil2 + 55;
            if((bil3>=0)&&(bil3<=9))
                        bil3=bil3 + 48;
            else
                        bil3=bil3 + 55;

                        hasilkonversi=adc2;
                        //konversi adc 0-3FF ke bentuk satuan
                        bil4=hasilkonversi/256;
                        hasilkonversi=hasilkonversi-256*bil4;
                        bil5=hasilkonversi/16;
                        bil6=hasilkonversi-16*bil5;

            if((bil4>=0)&&(bil4<=9))
                        bil4=bil4 + 48;
            else
                        bil4=bil4 + 55;
            if((bil5>=0)&&(bil5<=9))
                        bil5=bil5 + 48;
            else
                        bil5=bil5 + 55;
            if((bil6>=0)&&(bil6<=9))
                        bil6=bil6 + 48;
            else
                        bil6=bil6 + 55;
                        jam1=hour/16;
                        jam2=hour-16*jam1;
                        mnt1=minute/16;
                        mnt2=minute-16*mnt1;
                        for(i=0;i<512;i++)
                        {
                                if((sector[i]==0x0A)&&(sector[i+1]==0x00))    //cari tombol enter
                                {
                                i=i+1;
                                sector[j]=jam1+0x30;
                                sector[j+1]=jam2+0x30;
                                sector[j+2]=0x20;
                                sector[j+3]=mnt1+0x30;
                                sector[j+4]=mnt2+0x30;
                                sector[j+5]=0x20;
                                sector[j+6]='$';
                                sector[j+7]=bil1;
                                sector[j+8]=bil2;
                                sector[j+9]=bil3;
                                sector[j+10]=0x20;
                                sector[j+11]='$';
                                sector[j+12]=bil4;
```

```
                                sector[j+13]=bil5;
                                sector[j+14]=bil6;
                                sector[j+15]=0x0D;
                                sector[j+16]=0x0A;

                                i=512;
                                }
        else
         {       sector[i]=sector[i];}

          }
        //tulis data
        write_block(sect_clust_skrg);
}

void konversi_BCD_desimal(unsigned int BCD) //untuk penanggalan file packed time
{
        if (BCD<0x10)desimal=BCD;
        else if ((BCD>=0x10)&&(BCD<0x20))desimal=BCD-6;
        else if ((BCD>=0x20)&&(BCD<0x30))desimal=BCD-12;
        else if ((BCD>=0x30)&&(BCD<0x40))desimal=BCD-18;
        else if ((BCD>=0x40)&&(BCD<0x50))desimal=BCD-24;
        else if ((BCD>=0x50)&&(BCD<0x60))desimal=BCD-30;

}

unsigned int tulis_root(unsigned long alamat)
{int i,j,k,nomerfile,clust_LO_LSB,clust_LO_MSB;
char  tgl1,tgl2,bln1,bln2,th1,th2;
unsigned int packedtime,packeddate;
unsigned int temppackedtime,temppackeddate;

        //baca root entry taruh di sector[]
        /*
        cari root directory yg sectornya masi kosong dulu
                */
        for(i=0;i<=32;i++)
                {read_block(alamat+i);
                        for(j=0;j<=16;j++)
                        {
                        if ((sector[j]*32==0x00)||(sector[j]*32==0xE5))
                        {i=100;j=100;}
                        else alamat=alamat+i; }
                }

        read_block(alamat);
        clust_LO_LSB = clust_x;                         // ubah ke little endian
        clust_LO_MSB = clust_x >> 8;

        //ini buat packed time dan packed date
        konversi_BCD_desimal(hour);
        packedtime=desimal<<11;
        konversi_BCD_desimal(minute);
        desimal=desimal<<5;
        packedtime=packedtime+desimal;
        konversi_BCD_desimal(second);
        packedtime=packedtime+desimal;
        temppackedtime=packedtime<<8;                   //ubah ke little endian
        temppackedtime=temppackedtime/256;              //byar jadi 1 byte

        packedtime=packedtime>>8;

        //packed date
        konversi_BCD_desimal(year);
        desimal=desimal+2000-1980;
        packeddate=desimal<<9;
        konversi_BCD_desimal(month);
        desimal=desimal<<5;
        packeddate=packeddate+desimal;
        konversi_BCD_desimal(date);
        packeddate=packeddate+desimal;
        temppackeddate=packeddate<<8;                   //ubah ke little endian
        temppackeddate=temppackeddate/256;              //byar jadi 1 byte
        packeddate=packeddate>>8;

        //selesai

        tgl1=date/16;
        tgl2=date-16*tgl1;
        bln1=month/16;
        bln2=month-16*bln1;
        th1=year/16;
        th2=year-16*th1;
        //cari yg masi 00 atau E5
        for(i=0;i<511;i++)
                {
                        if((sector[i*32]==0x00)||(sector[i*32]==0xE5))
        {       //for(j=i*32;j<i*32+32;j++)
                j=i*32;
                sector[j]=tgl1+0x30;
                sector[j+1]=tgl2+0x30;
                sector[j+2]='_';
                sector[j+3]=bln1+0x30;
                sector[j+4]=bln2+0x30;
                sector[j+5]='_';
                sector[j+6]=th1+0x30;
                sector[j+7]=th2+0x30;
                sector[j+8]='T';
                sector[j+9]='X';
                sector[j+10]='T';
                sector[j+14]=temppackedtime;
                sector[j+15]=packedtime;
                sector[j+16]=temppackeddate;
                sector[j+17]=packeddate;
```

```
                sector[j+22]=temppackedtime;
                sector[j+23]=packedtime;
                sector[j+24]=temppackeddate;
                sector[j+25]=packeddate;
                sector[j+26]=clust_LO_LSB;
                sector[j+27]=clust_LO_MSB;
                sector[j+28]=0x00;
                sector[j+29]=0x40;
                i=512;}
        else
         {      for(k=i*32;k<i*32+32;k++)
                        sector[k]=sector[k];
                        nomerfile++
        }


         }
        //tulis sector[] ke root entry
        write_block(alamat);
        //tulis ke sector2 fisik
        for(i=0;i<512;i++)
            {sector[i]=0x00;
         }
        sector[0]=date;
        sector[1]=clust_LO_LSB;
        sector[2]=clust_LO_MSB;
        write_block_fisik(2);
}
unsigned int updatefat()
{int i,j;
unsigned long alamat;

        //alamat itu vbr + reserved sector, tapi vbr kan = 0
        alamat=reserved_sector;
        //baca FAT entry taruh di sector[]
        read_block(alamat);

        //cari yg masi 00
        for(i=0;i<512;i++)
   {
          if(sector[i]==0x00)
          {
                sector[i]=0xff;
                sector[i+1]=0xff;
                i=512;}
        else
         {      sector[i]=sector[i];}

         }
        //tulis sector[] ke root entry
        write_block(alamat);
        if(num_of_FAT == 2)
                write_block((unsigned long)reserved_sector + (unsigned long)sector_per_FAT);
        }
```

LAMPIRAN IV

LISTING PROGRAM

DELPHI

UNIT 1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, ExtCtrls, XPMan, ComCtrls, TeEngine, Series,
  TeeProcs, Chart;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    OpenDialog1: TOpenDialog;
    XPManifest1: TXPManifest;
    StatusBar1: TStatusBar;
    Button4: TButton;
    GroupBox1: TGroupBox;
    StringGrid1: TStringGrid;
    GroupBox2: TGroupBox;
    Chart1: TChart;
    Series1: TLineSeries;
    Button2: TButton;
    Button3: TButton;
    Chart2: TChart;
    Series2: TLineSeries;
    Edit2: TEdit;
    Label4: TLabel;
    Edit3: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Label7: TLabel;
    Edit6: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    Edit7: TEdit;
    StringGrid2: TStringGrid;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);


  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit2, Unit3, Unit4;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
  var
    myFile : TextFile;
    text,namafile   : string;
    jam,menit,t,p,i,nil1,nil2,nil3,nil4     : integer;
    rata1,rata2: real;
  begin
    opendialog1.Execute;
    // Try to open the Test.txt file for writing to
    namafile:=opendialog1.FileName;
    statusbar1.Panels[0].Text:=namafile;
    label3.Caption:=namafile;
    AssignFile(myFile, namafile);
    // Reopen the file for reading
    Reset(myFile);

  series1.Clear;
  series2.Clear;
  form2.series1.Clear;
  form3.Series2.Clear;

    i:=1;
    rata1:=0;
    rata2:=0;
    nil1:=0;
    nil2:=0;
    nil3:=4000;
    nil4:=4000;
   // Display the file contents
  while not Eof(myFile) do
   begin
    //  ReadLn(myFile, text);
      ReadLn(myFile, jam, menit, t, p);

      stringgrid1.RowCount:=i;
      if i>1 then
      StringGrid1.FixedRows:=1;
```

```
 edit1.Text:=inttostr(i-1);

  if t<>0 then        begin
  rata1:=rata1+t;
   if nil1>t then nil1:=nil1
  else nil1:=t;
   if nil3<t then  nil3:=nil3
  else  nil3:=t;     end;

   if p<>0 then       begin
  rata2:=rata2+p;
  if nil2>p then  nil2:=nil2
  else nil2:=p;
  if nil4<p then  nil4:=nil4
  else nil4:=p;       end;

 if Eof(myfile)  then
begin rata1:=rata1/(i-1);
  edit2.Text:=floattostr(rata1);
  rata2:=rata2/(i-1);
  edit5.Text:=floattostr(rata2);
  Edit3.Text:=inttostr(nil1);
  Edit6.Text:=inttostr(nil2);
  Edit4.Text:=inttostr(nil3);
  Edit7.Text:=inttostr(nil4);
  Chart1.LeftAxis.Automatic:=false;
  Chart2.LeftAxis.Automatic:=false;
  Chart1.LeftAxis.Maximum:=nil1+20;
  Chart2.LeftAxis.Maximum:=nil2+20;
  end;

  stringgrid1.Cells[0,i]:='  '+inttostr(i);
  stringgrid1.Cells[1,i]:='   '+inttostr(jam);
  stringgrid1.Cells[2,i]:='   '+inttostr(menit);
  stringgrid1.Cells[3,i]:='       '+inttohex(t,3);
  stringgrid1.Cells[4,i]:='       '+inttostr(t);
  stringgrid1.Cells[5,i]:='       '+inttohex(p,3);
  stringgrid1.Cells[6,i]:='       '+inttostr(p);


  i:=i+1;

   With series1 do
    Begin
    Add(  t, '' , clRed ) ;
    end;
  With Series2 do
    Begin
    Add(  p, '' , clRed ) ;
    end;

  With form2.Series1 do
    Begin
    Add(  t, '' , clRed ) ;
    end;
  With form3.Series2 do
    Begin
    Add(  p, '' , clRed ) ;
    end;
     end;

  // Close the file for the last time
  CloseFile(myFile);

end;

procedure TForm1.FormCreate(Sender: TObject);
begin
Chart1.LeftAxis.Automatic:=true;
Chart2.LeftAxis.Automatic:=true;
stringgrid1.ColWidths[0]:=30;
stringgrid1.ColWidths[1]:=50;
stringgrid1.ColWidths[2]:=50;
stringgrid1.ColWidths[3]:=80;
stringgrid1.ColWidths[4]:=80;
stringgrid1.ColWidths[5]:=80;
stringgrid1.ColWidths[6]:=80;
stringgrid1.Cells[0,0]:=' No';
stringgrid1.Cells[1,0]:='  Jam';
stringgrid1.Cells[2,0]:='  Menit';
stringgrid1.Cells[3,0]:='       Hex';
stringgrid1.Cells[4,0]:='       Dec';
stringgrid1.Cells[5,0]:='       Hex';
stringgrid1.Cells[6,0]:='       Dec';

stringgrid2.ColWidths[0]:=30;
stringgrid2.ColWidths[1]:=101;
stringgrid2.ColWidths[2]:=161;
stringgrid2.ColWidths[3]:=161;
stringgrid2.Cells[0,0]:=' ';
stringgrid2.Cells[1,0]:='          Waktu';
stringgrid2.Cells[2,0]:='                ADC 1';
stringgrid2.Cells[3,0]:='                ADC 2';

series1.Clear;
series2.Clear;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin

form2.showmodal;
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
form3.showmodal;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
aboutbox.showmodal;
end;

end.
```

UNIT 2

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, TeEngine, Series, ExtCtrls, TeeProcs, Chart, StdCtrls, XPMan;

type
  TForm2 = class(TForm)
    Chart1: TChart;
    Series1: TLineSeries;
    XPManifest1: TXPManifest;
    Button1: TButton;
    Label2: TLabel;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Chart1.LeftAxis.Automatic:=true;
Chart1.bottomAxis.Automatic:=true;
end;
```

```
end.

UNIT 3
unit Unit3;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, TeEngine, Series, ExtCtrls, TeeProcs, Chart, StdCtrls, XPMan;

type
  TForm3 = class(TForm)
    Chart2: TChart;
    Series2: TLineSeries;
    Label1: TLabel;
    XPManifest1: TXPManifest;
    Button1: TButton;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.Button1Click(Sender: TObject);
begin
  Chart2.LeftAxis.Automatic:=true;
Chart2.bottomAxis.Automatic:=true;
end;

end.
```

UNIT 4

```
unit Unit4;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, jpeg;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
```

```
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    OKButton: TButton;
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

{$R *.dfm}

end.
```

LAMPIRAN V

DATASHEET