Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 Dengan Antar Muka VNC1L

SKRIPSI
JURUSAN TKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

DWI OKTAVIANTO WAHYU NUGROHO
NIM. 021 063 0039 – 63

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS BRAWIJAYA
MALANG
2009

Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 Dengan Antar Muka VNC1L

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

DWI OKTAVIANTO WAHYU NUGROHO
NIM. 021 063 0039 – 63

Telah diperiksa dan disetujui oleh

DOSEN PEMBIMBING :

Pembimbing 1	Pembimbing 2
R. Arief Setyawan ST. MT.	Ir. Bambang Siswojo, MT.
NIP. 132 231 713	NIP. 131 759 588

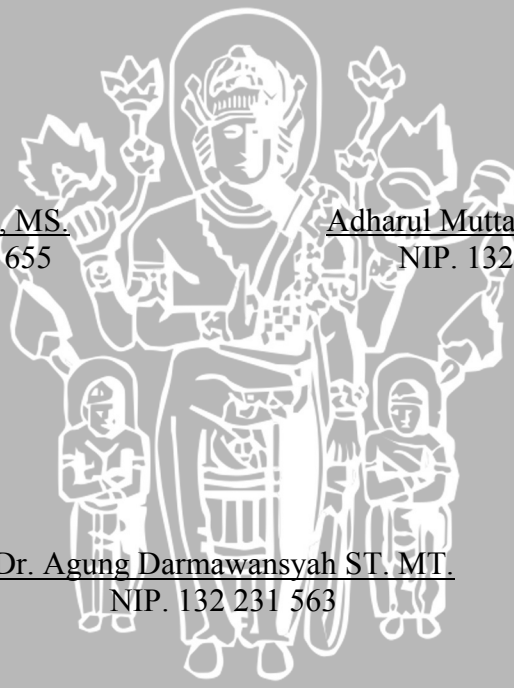
Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 Dengan Antar Muka VNC1L

Disusun oleh :

DWI OKTAVIANTO WAHYU NUGROHO
NIM. 021 063 0039 – 63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal **6 Februari 2009**

Majelis Penguji :



Ir. M. Julius St, MS.
NIP. 131 124 655

Adharul Muttaqin, ST, MT,
NIP. 132 311 88

Dr. Agung Darmawansyah ST. MT.
NIP. 132 231 563

Mengetahui :
Ketua Jurusan Teknik Elektro

Ir.Heru Nurwasito, M.Kom.
NIP. 131 879 033

PENGANTAR

Alhamdulillah, segala puji hanya bagi **Allah SWT** yang telah memberikan rahmat-Nya sehingga Penulis bisa menyelesaikan penulisan skripsi dengan judul **"Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 Dengan Antar Muka VNC1L"** ini. Tidak lupa, Penulis sampaikan terima kasih kepada semua pihak yang telah membantu dalam penulisan skripsi ini, baik berupa bantuan ide, bimbingan, referensi, materi, administrasi, dan dukungan semangat, antara lain kepada:

1. Bapak Ir. Heru Nurwarsito, MKom selaku Ketua Jurusan Teknik Elektro dan Bapak Rudy Yuwono, ST., M.Sc selaku Sekretaris Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya
2. Bapak Ir. M. Julius ST., MS dan Bapak Ir. Ponco Siswindarto, MS selaku Ketua Studi Konsentrasi Elektronika dan mantan Ketua Ketua Studi Konsentrasi Elektronika Teknik Elektro Universitas Brawijaya
3. Bapak R. Arief Setyawan, ST. MT. selaku pembimbing pertama dan Bapak Ir. Bambang Siswoyo, MT. selaku pembimbing kedua
4. Segenap karyawan pengajaran dan *recording* Teknik Elektro Universitas Brawijaya
5. Bapak Drs. H. L. Winarno A.W. dan Ibu Hj. Sutjiasri, serta keluarga tercinta yang telah memberikan dukungan yang tak ternilai
6. Teman – teman, terutama Agus Kurniawan., adik – adik team robot dan workshop TEUB, serta rekan – rekan angkatan 2002, 2003 dan 2004.
7. Dan semua pihak-pihak yang tidak bisa disebutkan satu – persatu

Semoga skripsi ini bermanfaat, baik bagi Penulis maupun bagi Pembaca. Saran dan kritik yang membangun akan Penulis terima dengan kerendahan hati.

Malang, Januari 2009

Penulis

ABSTRAKSI

Dwi Oktavianto Wahyu Nugroho, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, januari 2009, Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 Dengan Antar Muka VNC1L, Dosen pembimbing: R. Arief Setyawan, ST. MT. dan Ir. Bambang Siswoyo

Pada perkembangan teknologi saat ini, "*flash drive*" atau yang biasanya disebut *flash disk* telah menjadi suatu kebutuhan primer. Dengan ukuran yang kecil dan mudah dibawa serta harga yang relatif lebih murah menjadikan piranti ini menjadi pilihan banyak orang untuk menyimpan data secara *mobile*. Akan tetapi media ini memiliki kelemahan yakni dibutuhkannya sebuah komputer untuk pengaksesan data yang tersimpan dalam *flash drive* tersebut. Dengan kata lain untuk keperluan memindahkan data masih disibukkan dengan mencari komputer/laptop terlebih dahulu. Selain itu daya listrik yang digunakan oleh sebuah komputer/laptop untuk memindahkan data juga terbilang cukup besar dibanding dengan besarnya data yang mau dipindahkan.

Untuk mengatasi hal tersebut maka dibutuhkanlah suatu alat pengkopi data antar *flash drive* yang mampu dibawa kemana saja untuk memudahkan dalam hal pentransferan data dan lebih hemat dalam hal penggunaan energi.

Pada prinsipnya alat ini melakukan pembacaan data pada suatu *flash drive* yang ditampilkan kedalam lcd, melalui penekanan tombol tekan hingga data yang didinginkan tersebut ditemukan. Kemudian melakukan pentransferan atas data tersebut ke *flash drive* yang lain melalui penekan tombol "*copy*". Dengan demikian data akan terkopi dari *flash drive* satu ke *flash drive* yang lainnya tanpa perlu mencari sebuah komputer terlebih dahulu serta cenderung lebih hemat energi bila dibandingkan dengan media pengkopian data yang menggunakan komputer/laptop.

Pada alat pengkopi data antar *flash drive* ini memanfaatkan IC VNC1L–1A yang mampu bertindak sebagai USB *host* yang mampu menangani masalah protokol USB dan file sistem FAT. Sedangkan sebagai pengontrol IC VNC1L–1A digunakan ATmega64. Hasil dari pemrosesan data tersebut kemudian ditampilkan ke dalam LCD Nokia 3310.

Kata kunci : *flash drive*, pengkopian data, VNC1L-1A, ATmega64

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Lampiran

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada perkembangan teknologi saat ini, "*flash drive*" atau yang biasanya disebut *flash disk* telah menjadi suatu kebutuhan primer. Dengan ukuran yang kecil dan mudah dibawa serta harga yang relatif lebih murah menjadikan piranti ini menjadi pilihan banyak orang untuk menyimpan data secara *mobile*. Akan tetapi media ini memiliki kelemahan yakni dibutuhkannya sebuah komputer untuk pengaksesan data yang tersimpan dalam *flash drive* tersebut. Dengan kata lain untuk keperluan memindahkan data masih disibukkan dengan mencari komputer/laptop terlebih dahulu. Selain itu daya listrik yang digunakan oleh sebuah komputer/laptop untuk memindahkan data juga terbilang cukup besar dibanding dengan besarnya data yang akan dipindahkan.

Untuk mengatasi hal tersebut maka dibutuhkanlah suatu alat pengkopi data antar *flash drive* yang mampu dibawa kemana saja untuk memudahkan dalam hal pentransferan data dan lebih hemat dalam hal penggunaan energi.

Pada prinsipnya alat ini melakukan pembacaan data pada suatu *flash drive* yang ditampilkan kedalam lcd, melalui penekanan tombol tekan hingga data yang didinginkan tersebut ditemukan. Kemudian melakukan pentransferan atas data tersebut ke *flash drive* yang lain melalui penekan tombol "*copy*". Dengan demikian data akan terkopi dari *flash drive* satu ke *flash drive* yang lainnya tanpa perlu mencari sebuah komputer terlebih dahulu serta cenderung lebih hemat energi bila dibandingkan dengan media pengkopian data yang menggunakan komputer/laptop.

Pada alat pengkopi data antar *flash drive* ini memanfaatkan IC VNC1L–1A yang mampu bertindak sebagai USB *host* yang mampu menangani masalah protokol USB dan file sistem FAT. Sedangkan sebagai pengontrol IC VNC1L–1A digunakan ATmega64. Hasil dari pemrosesan data tersebut kemudian ditampilkan ke dalam LCD Nokia 3310.

1.2. Perumusan Masalah

Rumusan masalah perancangan alat pengkopi data antar *flash drive* ini adalah bagaimana merancang dan memanfaatkan Mikrokontroller ATmega64 dan VNC1L untuk membuat alat pengkopi data antar *flash drive* yang hemat listrik sehingga dapat mengurangi ketergantungan penggunaan komputer untuk melakukan pengkopian data.

1.3. Tujuan Karya

Penelitian ini bertujuan :

1.) Membuat alat pengkopi data antar *flash drive* menggunakan ATmega64 dan VNC1L
2.) Merancang alat pengkopi data antar *flash drive* yang portable dan ekonomis.

1.4. Kegunaan Karya

Kegunaan karya ini adalah membantu pelaksanaan proses pentransferan data, dalam hal ini dipusatkan pada masalah pengkopian data, antar *flash drive* yang lebih efektif dan efisien serta cenderung lebih hemat energi.

1.5. Ruang Lingkup

Dalam perancangan alat pengkopi data antar *flash drive* menggunakan ATmega64 dengan antar muka VNC1L ini permasalahan dibatasi dalam beberapa hal yaitu:

1.) Menekankan pada perancangan, pembuatan dan pembahasan sistem pengkopian data antar *flash drive* menggunakan ATmega64 dengan antar muka VNC1L
2.) Parameter keberhasilan alat pada kemampuan untuk melakukan proses pencarian data dan pengkopian data antar *flash drive.*
3.) Pembahasan ditekankan pada proses pengkopian data antar *flash drive* menggunakan ATmega64 dengan antar muka VNC1L.
4.) *Flash drive* yang digunakan adalah *flash drive* dengan tipe FAT32.

1.6. Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan tugas akhir ini adalah sebagai berikut:

Bab I : Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, ruang lingkup pembahasan dan sistematika penulisan.

Bab II : Tinjauan Pustaka

Membahas teori-teori dasar dari beberapa piranti dan metode yang digunakan dalam menunjang perancangan dan pembuatan alat.

Bab III : Metode Penelitian

Menjelaskan tahap-tahap dan metode yang dilakukan dalam perencanaan, pembuatan sampai dengan pengujian alat.

Bab IV : Perancangan dan Pembuatan

Memuat spesifikasi, diagram blok, prinsip kerja, perancangan dan pembuatan alat.

Bab V : Hasil dan Pembahasan

Memuat hasil pengujian alat dan analisis terhadap data hasil pengujian menggunakan teori yang ada.

Bab VI : Kesimpulan dan Saran

Berisi kesimpulan yang dapat diambil dan saran terhadap hasil yang diperoleh dalam tugas akhir ini.

BAB II
TINJAUAN PUSTAKA

Untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar-dasar perencanaan alat ini, maka perlu penjelasan dan uraian teori penunjang yang digunakan dalam penulisan tugas akhir ini.

Teori-teori penunjang yang dijelaskan dalam bab ini adalah :

1.) Mikrokontroller ATmega64
2.) VNC1L
3.) USB *Flash Drive*
4.) LCD (*Liquid Crystal Display*) Nokia 3310

2.1 Mikrokontroler ATmega 64

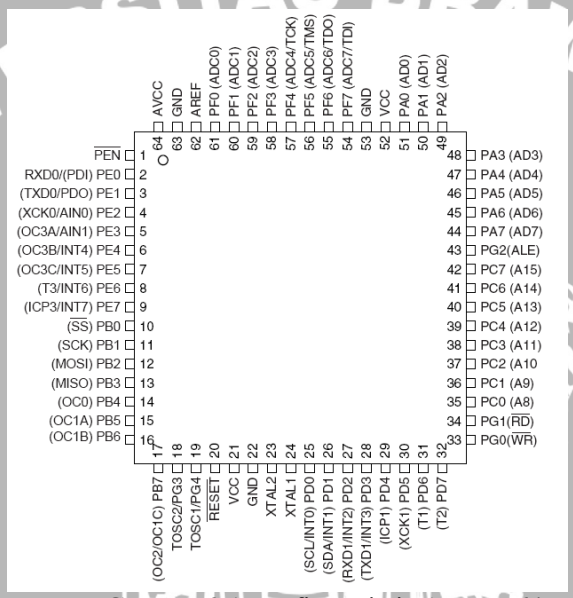
Mikrokontroller ATmega64 adalah sebuah mikrokontroller CMOS 8-bit performa tinggi yang hemat daya melalui pengembangan arsitektur RISC dengan 64K bytes I*n-System Reprogrammable Flash* dan 2 Kbytes EEPROM dengan daya tahan 10,000 siklus penulisan/penghapusan serta dilengkapi dengan 4 Kbyte internal SRAM. Mikrokontroller ini dibuat menggunakan teknologi *high-density nonvolatile memory* milik Atmel dan kompatibel dengan standar industri sekumpulan instruksi and pin-out AVR. *On-chip downloadable Flash* memungkinkan memori program untuk diprogram ulang di dalam sistem melalui sebuah antarmuka serial SPI atau dengan sebuah program memori nonvolatile yang konvensional. Dengan kemampuan eksekusi *powerfull instruction* dalam satu siklus waktu, sehingga ATmega mampu mencapai 1 MIPS per MHZ, yang memberikan kesempatan perancang sistem untuk mengoptimalkan konsumsi daya dengan tetap mempertimbangkan kecepatan pemrosesan.

Mikrokontroller ATmega64 memiliki kelengkapan sebagai berikut:

- Memiliki 130 instruksi yang *powerful*
- Kapasitas *Flash* memori yang *Downloadable* sebesar 8 Kbyte
- Kapasitas EEPROM sebesar 2 Kbyte
- Memiliki 8 buah kanal ADC, 10-bit dengan penguatan (1x, 10x, 200x)

- Kapasitas RAM internal sebesar 4 Kbyte
- Jalur I/O berjumlah 53 buah
- Dua buah *programmable* USART (serial port)
- *Master*/*Slave* SPI serial *interface*
- Dapat mencapai hingga 16 MIPS pada frekuensi kerja 16 MHz
- Tegangan operasi antara 4,5 volt sampai 5,5 volt.

Konfigurasi pin mikrokontroler ATmega64 dapat dilihat dalam Gambar 2.1.



Gambar 2.1 Konfigurasi pin ATmega64
Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

2.1.1 Struktur dan Operasi Port

Mikrokontroler ATmega64 memiliki 7 buah port I/O. Enam buah port memiliki 8 buah jalur I/O dan sebuah port memiliki 5 buah jalur I/O. Beberapa karakteristik port ATmega64 dijelaskan secara singkat berikut ini :

1.) Setiap jalur I/O memiliki buffer, penahan (*latch*), kemudi *input* dan *output*.

2.) Setiap jalur I/O terdapat register pengatur guna dijadikan *input* atau *output*.

3.) Port A (PA7..PA0)

Port A merupakan port I/O 8 bit *bi-directional* dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Port A juga mempunyai fungsi khusus seperti yang dijabarkan dalam Tabel 2.1.

Tabel 2.1 Fungsi alternatif pin port A

Port Pin	Fungsi alternatif
PA7	AD7(bit ke-7 data dan alamat antarmuka memori eksternal)
PA6	AD6(bit ke-6 data dan alamat antarmuka memori eksternal)
PA5	AD5(bit ke-5 data dan alamat antarmuka memori eksternal)
PA4	AD4(bit ke-4 data dan alamat antarmuka memori eksternal)
PA3	AD3(bit ke-3 data dan alamat antarmuka memori eksternal)
PA2	AD2(bit ke-2 data dan alamat antarmuka memori eksternal)
PA1	AD1(bit ke-1 data dan alamat antarmuka memori eksternal)
PA0	AD0(bit ke-0 data dan alamat antarmuka memori eksternal)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

4.) Port B (PB7..PB0)

Port B merupakan port I/O 8 bit *bi-directional* dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Port B juga mempunyai fungsi khusus seperti yang dijabarkan dalam Tabel 2.2.

Tabel 2.2 Fungsi alternatif pin port B

Port Pin	Fungsi alternative
PB7	OC2/OC1C (pembanding keluaran dan keluaran PWM untuk *Timer*/*Counter*2 atau pembanding keluaran dan keluaran PWM C untuk *Timer*/*Counter*1)
PB6	OC1B (pembanding keluaran dan keluaran PWM B untuk *Timer*/*Counter*1)
PB5	OC1A (pembanding keluaran dan keluaran PWM A untuk *Timer*/*Counter*1)
PB4	OC0 (pembanding keluaran dan keluaran PWM untuk *Timer*/*Counter*0)
PB3	MISO (SPI *Bus Master Input/Slave Output*)
PB2	MOSI (SPI *Bus Master Output/Slave Input*)
PB1	SCK(SPI *Bus Serial Clock*)
PB0	SS (SPI *Slave Select Input*)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

5.) Port C (PC7..PC0)

Port C merupakan port I/O 8 bit *bi-directional* dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Port C mempunyai fungsi khusus seperti yang dijabarkan dalam Tabel 2.3.

Tabel 2.3 Fungsi alternatif pin port C

Port Pin	Fungsi alternatif
PC7	A15
PC6	A14
PC5	A13
PC4	A12
PC3	A11
PC2	A10
PC1	A9
PC0	A8

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

6.) Port D (PD7..PD0)

Port D merupakan port I/O 8 bit *bi-directional* dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Port D mempunyai fungsi khusus seperti yang dijabarkan dalam Tabel 2.4.

Tabel 2.4 Fungsi alternatif pin port D

Port Pin	Fungsi alternatif
PD7	T2(masukan pewaktu *Timer*/*Counter*2)
PD6	T1(masukan pewaktu *Timer*/*Counter*1)
PD5	XCK1(masukan/keluaran pewaktu USART1 eksternal)
PD4	IC1(pencuplik *trigger* masukan *Timer*/*Counter*1)
PD3	INT3/TXD1(masukan *interupt*3 eksternal atau pin pengirim UART1)
PD2	INT2/RXD1(masukan *interupt*2 eksternal atau pin penerima UART1)
PD1	INT1/SDA(masukan *interupt*1 eksternal atau data serial TWI)
PD0	INT0/SDA(masukan *interupt*0 eksternal atau pewaktu serial TWI)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

7.) Port E (PE7..PE0)

Port E merupakan port I/O 8 bit dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Port E mempunyai fungsi khusus seperti yang dijabarkan dalam Tabel 2.5.

Tabel 2.5 Fungsi alternatif pin port E

Port Pin	Fungsi alternatif
PE7	INT7/IC3(masukan *interupt*7 eksternal atau pencuplik *trigger* masukan *Timer*/*Counter*3)
PE6	INT6/T3(masukan *interupt*6 eksternal atau pewaktu masukan *Timer*/*Counter*3)
PE5	INT5/OC3C(masukan *interupt*5 eksternal atau pembanding keluaran dan keluaran PWM C untuk *Timer*/*Counter*3)
PE4	INT4/OC3B(masukan *interupt*5 eksternal atau pembanding keluaran dan keluaran PWM B untuk *Timer*/*Counter*3)
PE3	AIN1/OC3A(pembanding masukan negatif analog atau pembanding keluaran dan keluaran PWM A untuk *Timer*/*Counter*3)
PE2	AIN0/XCK0(pembanding masukan positif analog atau masukan/keluaran pewaktu USART0 eksternal)
PE1	PDO/TXD0(keluaran data pemrograman atau pin pengirim UART0)
PE0	PDI/RXD0(masukan data pemrograman atau pin penerima UART0)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

8.) Port F (PF7..PF0)

Port F merupakan port I/O 8 bit dengan resistor *pull-up* (yang dapat dipilih untuk tiap-tiap bit). Selain sebagai I/O, Fungsi khusus Port F yang dapat digunakan adalah seperti yang dijabarkan dalam Tabel 2.6.

Tabel 2.6 Fungsi alternatif pin port F

Port Pin	Fungsi alternatif
PF7	ADC7/TDI(masukan ADC channel ke-7 atau masukan data tes JTAG)
PF6	ADC6/TDO(masukan ADC channel ke-6 atau keluaran data tes JTAG)
PF5	ADC5/TMS(masukan ADC channel ke-5 atau mode pemilihan tes JTAG)
PF4	ADC4/TCK(masukan ADC channel ke-4 atau pewaktu tes JTAG)
PF3	ADC3(masukan ADC channel ke-3)
PF2	ADC3(masukan ADC channel ke-2)
PF1	ADC3(masukan ADC channel ke-1)
PF0	ADC3(masukan ADC channel ke-0)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

9.) Port G (PG4..PG0)

Port G merupakan port I/O 5 bit dengan resistor *pull-up* (yang mampu dipilih untuk masing-masing bit). Selain sebagai I/O Port G juga dapat digunakan sebagai fungsi khusus seperti yang dijabarkan dalam Tabel 2.7.

Tabel 2.7 Fungsi alternatif pin port G

Port Pin	Fungsi alternatif
PG4	TOSC1(RTC *Oscillator Timer*/*Counter*0)
PG3	TOSC2(RTC *Oscillator Timer*/*Counter*0)
PG2	ALE(*Address Latch Enable* untuk memori eksternal)
PG1	RD(*Read strobe* untuk memori eksternal)
PG0	WR(*write strobe* untuk memori eksternal)

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

2.1.2 Sistem interupt

Mikrokontroller ATmega64 memiliki 35 alamat vektor interupt dan reset dimana nomor urut dari vector interupt tadi menyatakan prioritas dari tinterupt tersebut. Alamat vector interupt dari mikrokontroller ATmega64 dapat dilihat dalam Tabel 2.8.

Tabel 2.8 Alamat vector interupt dan reset dari ATmega64

No. vektor	Alamat program	Sumber	Definisi interupsi	No. vektor	Alamat program	Sumber	Definisi Interupsi
1	0x0000	RESET	Pin eksternal, *Power-on Reset*, *Brown-out Reset*, *Watchdog Reset* dan JTAG AVR Reset	18	0x0022	SPI STC	SPI *Serial Transfer Complete*
				19	0x0024	USART0 RX	USART0 *Rx Complete*
2	0x0002	INT0	Interupt eksternal 0	20	0x0026	USART0 UDRE	USART0 *Data Register Empty*
3	0x0004	INT1	Interupt eksternal 1	21	0x0028	USART0 TX	USART0 *Tx Complete*
4	0x0006	INT2	Interupt eksternal 2	22	0x002A	ADC	ADC *Convertion Complete*
5	0x0008	INT3	Interupt eksternal 3	23	0x002C	EE READY	EEPROM siap
6	0x000A	INT4	Interupt eksternal 4	24	0x002E	ANALOG COMP	Pembanding Analog
7	0x000C	INT5	Interupt eksternal 5	25	0x0030	TIMER1 COMPC	*Timer/Counter1 Compare Match C*
8	0x000E	INT6	Interupt eksternal 6	26	0x0032	TIMER3 CAPT	*Timer/Counter3 Capture Event*
9	0x0010	INT7	Interupt eksternal 7	27	0x0034	TIMER3 COMPA	*Timer/Counter3 Compare Match A*
10	0x0012	TIMER2 COMP	*Timer/Counter2 Compare Match*	28	0x0036	TIMER3 COMPB	*Timer/Counter3 Compare Match B*
11	0x0014	TIMER2 OVF	*Timer/Counter2 Overflow*	29	0x0038	TIMER3 COMPC	*Timer/Counter3 Compare Match C*
12	0x0016	TIMER1 CAPT	*Timer/Counter1 Capture Event*	30	0x003A	TIMER3 OVF	*Timer/Counter3 Overflow*
13	0x0018	TIMER1 COMPA	*Timer/Counter1 Compare Match A*	31	0x003C	USART1 RX	USART1 *Rx Complete*
14	0x001A	TIMER1 COMPB	*Timer/Counter1 Compare Match B*	32	0x003E	USART1 UDRE	USART1 *Data Register Empty*
15	0x001C	TIMER1 OVF	*Timer/Counter1 Overflow*	33	0x0040	USART1 TX	USART1 *Tx Complete*
16	0x001E	TIMER0 COMP	*Timer/Counter0 Compare Match*	34	0x0042	TWI	*Two-wire Serial Interface*
17	0x0020	TIMER0 OVF	*Timer/Counter0 Overflow*	35	0x0044	SPM READY	*Slave Program Memory Ready*

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

2.1.3 Komunikasi Serial USART(*Universal Syncronous and Asyncronous serial Receiver and Transmitter*)

Mikrokontroller ATmega64 dilengkapi dengan fasilitas komunikasi serial USART dengan spesifikasi sebagai berikut:

- Komunikasi full-duplex dengan register serial untuk penerima dan pengirim data.
- Dapat dioperasikan pada mode komunikasi sinkronus dan asinkronus
- Mempunyai resolusi tinggi untuk generator baudrate
- Layanan pengiriman data terdiri dari 5,6,7,8, dan 9 bit dan 1 atau 2 bit stop.
- Peritas genap atau ganjil dan didukung dengan pengecekan paritas oleh hardware.
- Memiliki filter noise yang terdiri dari pendeteksi kesalahan bit start dan low pass filter.
- Memiliki 3 layanan *interupt* yaitu TX *complite*, TX *data empty*, dan RX complite.
- Mode komunikasi multi processor
- Mode komunikasi asinkron dengan dua kecepatan.

Untuk menghitung boudrate dari komunikasi serial digunakan rumus seperti yang terlihat dalam Tabel 2.9.

Tabel 2.9 Rumus menghitung *baudrate*

Mode Operasi	Rumus perhitungan *Baud rate*	Rumus perhitungan nilai UBRR
Asinkronus mode normal (U2X=0)	$BAUD = \frac{fosc}{16(UBRR+1)}$	$UBRR = \frac{fosc}{16BAUD} - 1$
Asinkronus mode kecepatan ganda (U2X=1)	$BAUD = \frac{fosc}{8(UBRR+1)}$	$UBRR = \frac{fosc}{8BAUD} - 1$
Sinkronus mode master	$BAUD = \frac{fosc}{2(UBRR+1)}$	$UBRR = \frac{fosc}{2BAUD} - 1$

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

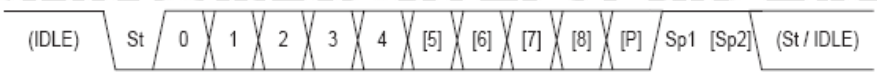
dengan :

f_{OSC} = Frekuensi *clock* dari sistem osilator

UBRR = Register *baudrate* yang terdiri dari UBRRL dan UBRRH

BAUD = *Baudrate* dalam bit per detik(bps)

Pada pengiriman data secara serial menggunakan ATmega64 memakai format seperti dalam Gambar 2.2.



Gambar 2.2 Format data pengiriman
Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

dengan :

St = Bit start selalu berlogika rendah

(n) = Banyaknya data yang dikirim (0-8)

P = Bit paritas (ganjil atau genap)

Sp = Bit stop selalu berlogika tinggi (bit stop bisa berjumlah 1 atau 2)

IDLE = Tidak ada data yang ditransfer pada RX dan TX, IDLE selalu berlogika tinggi

2.1.4 Komunikasi SPI (*Serial Pheripheral Interface*)

Mikrokontroller ATmega64 dilengkapi dengan fasilitas komunikasi *Serial Pheripheral Interface* yang biasa disingkat dengan SPI dengan fiture sebagai berikut:

- Komunikasi full-duplex, dengan tiga jalur data transfer sinkronus
- Dapat dioperasikan sebagai Master atau Slave
- Data transfer yang dapat disetel, LSB dulu atau MSB dulu
- Tujuh bit rate yang dapat deprogram
- Flag interupsi apabila transmisi data berakhir
- Flag proteksi untuk kegagalan penulisan
- Sebagai pemicu berakhirnya mode Idle (*wake-up from Idle Mode*)
- Dua kali kecepatan mode SPI Master

Komunikasi SPI membutuhkan 4 jalur sinyal, yaitu:

- SCK (*Serial Clock*); yaitu sinyal *clock* yang menggeser bit yang hendak dituliskan ke dalam register geser terima AVR lain, dan menggeser bit yang hendak dibaca dari register geser kirim AVR lain. Untuk menentukan besarnya nilai SCK yang akan digunakan dilakukan dengan cara melakukan penyetingan konfigurasi bit SPI2X pada SPSR (*SPI Status Register*) serta bit SPR1 dan SPR2 pada SPCR (*SPI Control Register*).

Format konfigurasi dari SPI2X, SPR1, dan SPR2 untuk menentukan besarnya frekuensi SCK, terlihat dalam Tabel 2.10

Tabel 2.10 Format konfigurasi SPI2X, SPR1, dan SPR2

SPI2X	SPR1	SPR0	Frekuensi SCK
0	0	0	fosc/4
0	0	1	fosc/16
0	1	0	fosc/64
0	1	1	fosc/128
1	0	0	fosc/2
1	0	1	fosc/8
1	1	0	fosc/32
1	1	1	fosc/64

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

Untuk proses penyuntingan data serial pada SPI dilakukan dengan melakukan penyetingan bit SPOL dan SPHA pada SPCR, seperti yang terlihat dalam Tabel 2.11

Tabel 2.11 Konfigurasi bit SPOL dan SPHA pada SPCR

	Tepi naik	Tepi turun	Mode SPI
CPOL = 0, CPHA = 0	*Sample* (naik)	*Setup* (turun)	0
CPOL = 0, CPHA = 1	*Setup* (naik)	*Sample* (turun)	1
CPOL = 1, CPHA = 0	*Sample* (turun)	*Setup* (naik)	2
CPOL = 1, CPHA = 1	*Setup* (turun)	*Sample* (naik)	3

Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

- MOSI (*Master Out Slave In*); sinyal bit data serial yang hendak dituliskan dari master ke slave.
- MISO (*Master In Slave Out*); sinyal bit data serial yang hendak dibaca dari slave ke master.
- SS' (*Slave Select*/aktif rendah); sinyal untuk memilih dan mengaktifkan *slave*.

Saat menggunakan komunikasi SPI, mikrokontroller dapat disetel sebagai salah satu dari dua mode komuinikasi SPI yakni, *Master mode* atau *Slave mode*. Masing-masing mode tersebut memiliki karakteristik sebagai berikut:

- *Slave Mode*

  Saat SPI dikonfigurasikan sebagai sebuah *Slave*, pin *Slave Select* (SS) hanya berfungsi sebagai masukan. SPI akan aktif saat pin SS dikenai sinyal *low*, dan MISO akan berperan sebagai keluaran jika sudah dikonfigurasikan sebelumnya oleh pengguna. Pin yang lain berperan sebagai masukan. Ketika SS berubah menjadi *high*, semua pin berperan

sebagai input, dan SPI menjadi pasif, yang berarti bahwa perangkat tidak akan menerima data yang masuk.

Saat pin SS dikenai sinyal *high*, SPI *Slave* akan langsung me-*reset* logika yang dikirim dan diterimanya, dan menempatkan sebagian data yang diterima ke dalam register geser (*Shift Register*).

- *Master Mode*

  Ketika SPI dikonfighurasikan sebagaisebauh Master {MSTR dalam SPCR (*SPI Control Register*) di-*set*}, pengguna dapat menetukan arah dari pin SS.

  Jika SS dikonfigurasikan sebagai keluaran, pin akan berperan sebagai *general output* yang tidak akan memberikan dampak pada sistem SPI. Umumnya, pin akan mengendalikan pin SS pada SPI *Slave*.

  Jika SS dikonfigurasikan sebagai masukan, maka konsekuensinya harus menjaganya agar tetap berlogika tinggi yang bertujuan untuk menjaga operasi. Jika SS diberi logika rendah oleh rangkaian perangkat saat SPI dikonfigurasikan sebagai *Master* dengan pin SS yang berfungsi sebagai sebuah masukan, sistem SPI akan mengira bahwa Master yang lain telah memilih SPI sebagai Slave dan siap untuk mulai mengirim data. Untuk menghindari pertentangan jalur, sistem SPI melakukan beberapa tindakan berikut:

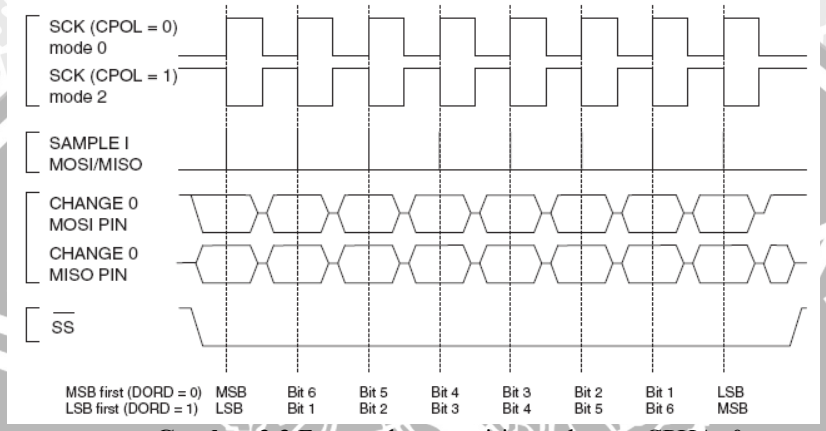
  1.) MSTR bit pada SPCR di-*clear* dan sistem SPI menjadi sebuah *Slave*. Sebagai akibatnya SPI menjadi sebuah *Slave*, pin MOSI dan SCK menjadi masukan.

  2.) Bendera SPIF (*SPI Interrupt Flag*) pada SPSR (*SPI Status Register*) di-*set*, dan jika SPI *interupt* di-*enable*, serta bit I pada SREG di-*set*, rutin *interupt* akan dijalankan.

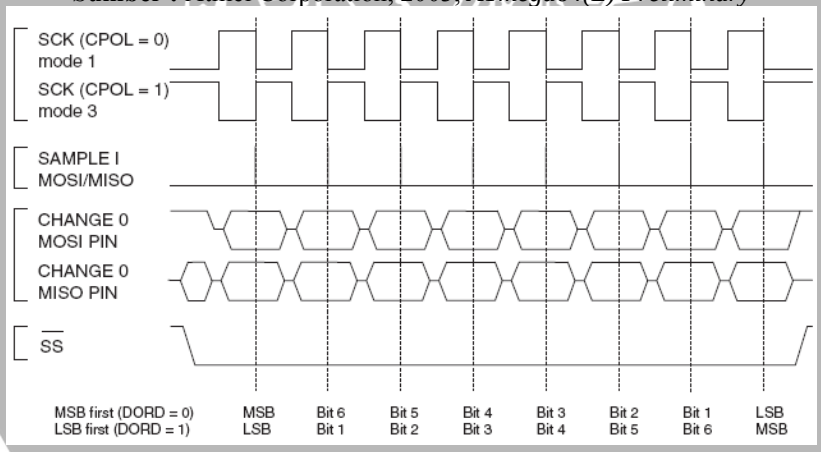
  sehingga, ketika kendali *interupt* transmisi SPI digunakan saat *Master Mode*, dan terjadi kemungkinan SS dikenai logika rendah, *interupt* harus selalu mengecek apakah bit MSTR masih di-*set*. Jika bit MSTR telah di-

clear oleh *slave select*, maka harus di-set oleh pengguna untuk mengaktifkan kembali SPI *Master mode*.

Pada pengiriman data SPI menggunakan ATmega64 memakai format seperti dalam Gambar 2.3 dan Gambar 2.4. Dimana penentuan bit pengiriman, yakni apakah mengirim dengan MSB atau LSB terlebih dahulu, dilakukan dengan melakukan konfigurasi pada bit DORD pada SPCR.



Gambar 2.3 Format data pengiriman dengan CPHA=0
Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*



Gambar 2.4 Format data pengiriman dengan CPHA=1
Sumber : Atmel Corporation, 2003, *ATmega64(L) Preliminary*

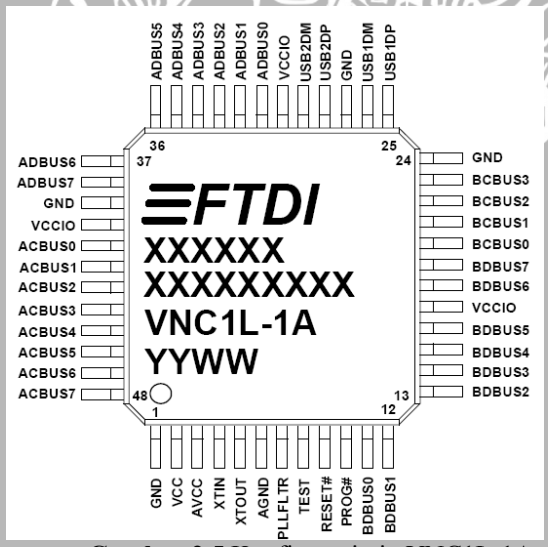
2.2 VNC1L

Vinculum VNC1L merupakan keluarga Vinculum FTDI pertama yang dilengkapi dengan USB *host controller* yang sudah diintegrasikan kedalam rangkaian piranti. Selain mampu menangani USB *Host Interface*, fungsi pentransferan data, mampu mengatur *inbuilt* MCU seperti yang diinginkan serta

dilengkapi dengan *Flash memory*. Vinculum juga mampu mengenali USB *device class*. Selain itu juga mampu menangani FAT saat diantarmukakan dengan piranti *mass-storage* seperti *flash drive*.

Vinculum VNC1L memiliki kelengkapan sebagai berikut:

- Penanganan USB protocol dalam chip
- 8 / 32 bit V-MCUCore
- *Clock multiplier* yang terintegrasi mencapai 12 MHz hingga 48 MHz.
- 64 kbyte embedded Flash ROM program memory
- 4 kbyte internal data SRAM
- Pemrograman *firmware* via USB *Flash drive* atau UART *interface*
- USB full speed 12 Mbps and low speed 1.5 Mbps
- USB host and slave device compatible
- Tegangan operasi 3.3 V dengan tegangan masukan 5 V.
- Antar-muka MCU / PLD / FPGA via UART, FIFO, or SPI interface



Gambar 2.5 Konfigurasi pin VNC1L-1A

Sumber : Future Technology Devices International Ltd, 2006, *Vinculum VNC1L Embedded USB Host Controller I.C.* Datasheet Version 0.95

Pemilihan jalur komunikasi dilakukan dengan cara memberikan resistor *pull-up* atau *pull-down* pada pin 46 dan pin 47, besarnya nilai resistor yakni sekitar 47 kΩ. Konfigurasi resistor tersebut dijabarkan dalam Tabel 2.12

Tabel 2.12 Konfigurasi resistor pada pin 46 dan 47

ACBUS6 (VNC1L pin 47)	ACBUS5 (VNC1L pin 46)	Mode I/O
Pull – Up	*Pull – Up*	Serial UART
Pull – Up	*Pull – Down*	SPI
Pull – Down	*Pull – Up*	Parallel FIFO
Pull – Down	*Pull – Down*	Serial UART

Sumber : Future Technology Devices International Ltd, 2006, *Vinculum VNC1L Embedded USB Host Controller I.C. Datasheet Version 0.95*

Setelah konfigurasi resistor telah ditentukan maka konfigurasi jalur komunikasi dijabarkan dalam Tabel 2.13

Tabel 2.13 Konfigurasi pemilihan bus

Pin	Nama	Port	Deskripsi	Konfigurasi pemilihan bus kontrol dan data				Pin	Nama	Port	Deskripsi	Konfigurasi pemilihan bus kontrol dan data			
				UART	Paralel FIFO	SPI slave	Port I/O					UART	Paralel FIFO	SPI slave	Port I/O
31	ADBUS0	I/O	Bus data/control bidireksional hingga 5V	TXD	D0	SCLK	AD0	37	ADBUS6	I/O	Bus data/control bidireksional hingga 5V	DCD#	D6		AD6
32	ADBUS1	I/O	Bus data/control bidireksional hingga 5V	RXD	D1	SDI	AD1	38	ADBUS7	I/O	Bus data/control bidireksional hingga 5V	RI#	D7		AD7
33	ADBUS2	I/O	Bus data/control bidireksional hingga 5V	RTS#	D2	SDO	AD2	39	ACBUS0	I/O	Bus data/control bidireksional hingga 5V	TXDEN#	RXF#		AC0
34	ADBUS3	I/O	Bus data/control bidireksional hingga 5V	CTS#	D3	CS	AD3	40	ACBUS1	I/O	Bus data/control bidireksional hingga 5V		TXF#		AC1
35	ADBUS4	I/O	Bus data/control bidireksional hingga 5V	DTR#	D4		AD4	41	ACBUS2	I/O	Bus data/control bidireksional hingga 5V		RD#		AC2
36	ADBUS5	I/O	Bus data/control bidireksional hingga 5V	DSR#	D5		AD5	42	ACBUS3	I/O	Bus data/control bidireksional hingga 5V		WR		AC3

Sumber : Future Technology Devices International Ltd, 2006, *Vinculum VNC1L Embedded USB Host Controller I.C. Datasheet Version 0.95*

2.2.1 File Sistem FAT

Dalam media penyimpan (*Storage Media*) file tidak serta merta disimpan dalam sebuah alamat, namun file tersebut dipecah menjadi sejumlah data yang tersimpan dalam beberapa alamat secara acak, sehingga untuk pengaksesan data tersebut diperlukan sebuah teknik yang mengaitkan antara data yang satu dengan data yang berikutnya. Dengan demikian data akan terbaca secara utuh dan berurutan. Teknik tersebut menggunakan *table link list* yang salah satunya adalah FAT (*File Allocation Table*). Terdapat tiga tipe FAT yang saat ini digunakan yakni, FAT 12, FAT 16, dan FAT 32. Perbedaan dari ke-tiga tipe FAT ini terletak pada jumlah bit yang digunakan untuk mengalamati data yang

disimpannya, dimana jumlah bit yang digunakan ditandai dengan besarnya angka yang digunakan dalam penamaan FAT itu sendiri (FAT 12 = 12 bit alamat, FAT 16 = 16 bit alamat, dan FAT 32 = 32 bit alamat).

Dalam FAT data disimpan dalam beberapa *sector* (dengan kapasitas 512 byte) yang terangkum menjadi sebuah *cluster*. Penyimpanan tersebut ditujukan untuk memudahkan penyimpanan file yang memerlukan kapasitas simpanan yang besar. Jumlah maksimum *cluster* yang digunakan berbeda, tergantung dari jenis FAT yang digunakan, dan besarnya ukuran *cluster* menentukan kapasitas maksimum media penyimpan.

Tabel 2.14 Tabel Jumlah maksimum cluster pada tiap-tiap jenis FAT

Ukuran FAT dalam bit	Jumlah cluster
12	4096
16	65536
32	4294967296

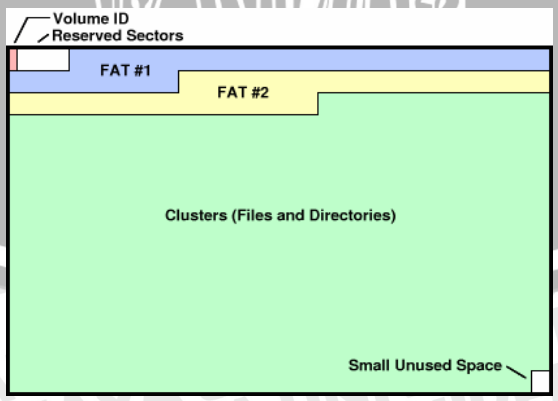
Sumber :. Heybruck, William F., Dr., 2005, *An Introduction to FAT 16/FAT 32 File Systems*, Hitachi Global Storage Technologies, Charlotte, NC

Tabel 2.15 Tabel ukuran cluster dan kapasitas maksimum media penyimpan

Ukuran *cluster* (byte)	FAT12	FAT16	FAT32
1024 (1K)	4MB	64MB	4TB
4096 (4K)	16MB	256MB	16TB
8192 (8K)	32MB	512MB	32TB
16384 (16K)	64MB	1GB	64TB
32768 (32K)	128MB	2GB	128TB

Sumber : Heybruck, William F., Dr., 2005, *An Introduction to FAT 16/FAT 32 File Systems*, Hitachi Global Storage Technologies, Charlotte, NC

Isi sistem *file* FAT tersusun atas empat bagian dasar yaitu daerah *Reserved*, daerah FAT, daerah *Root Director*, serta daerah *File* dan *Directory* data, seperti yang terlihat dalam Gambar2.6.



Gambar 2.6 Isi sistem file FAT32

Sumber : Stoffregen Paul, 2005,*Understanding FAT32 Filesystems*, PJRC

Dalam file sistem FAT 32, sektor pertama selalu *Volume ID*, diikuti dengan ruang yang tak terpakai/kosong yang disebut dengan *reserved sector*. Kemudian setelah *reserved sector*, adalah dua buah kopi FAT (*File Allocation Table*). Bagian terakhir dari file sistem adalah data yang tersusun dalam "*clusters*", dengan kemungkinan sedikit sisa ruang kosong setelah *cluster* terakhir.

Sebagian besar ruang *disk* merupakan daerah *cluster*, yang digunakan untuk menyimpan semua *file* dan direktori. *Cluster* dimulai dengan nomor 2, dengan kata lain data tidak akan mungkin disimpan dalam *cluster* #0 atau *cluster* #1. Hal ini dikarenakan *cluster* #0 dan *cluster* #1 digunakan oleh file sistem FAT sebagai penanda, yakni *cluster* #0 sebagai tanda bahwa *cluster* tersebut tidak sedang tidak digunakan (*unused*), dan *cluster* #1 sebagai tanda bahwa *cluster* tersebut digunakan (*occupied*).

Semua informasi fisik mengenai sistem file FAT didapatkan dari *Volume ID*, dimana isi dari *Volume ID* adalah seperti yang tercantum dalam Tabel 2.16.

Tabel 2.16 Volume ID pada FAT 32

Isi	Penamaan dalam microsoft	Offset	Ukuran	Nilai
Bytes per sector	BPB_BytsPerSec	0x0B	16 Bit	Selalu 512 bytes
Sector per cluster	BPB_SecPerClus	0x0D	8 Bit	1, 2, 4, 8, 16, 32, 64, 128
Jumlah *reserved sector*	BPB_RsvdSecCnt	0x0E	16 Bit	Umumnya 0x20
Jumlah FAT	BPB_NumFATs	0x10	8 Bit	Selalu 2
Sektor per FAT	BPB_FATSz32	0x24	32 Bit	Tergantung pada ukuran disk
Cluster pertama *root* direktori	BPB_RootClus	0x2C	32 Bit	Biasanya 0 x 0000 0002
Signature	(none)	0x1FE	16 Bit	Selalu 0 x AA55

Sumber : Stoffregen Paul, 2005,*Understanding FAT32 Filesystems*, PJRC

Seperti yang telah dijelaskan sebelumnya pengaksesan data dalam suatu media penyimpan dilakukan secara berantai. Oleh karena itu sangat penting untuk mengetahui letak *cluster* awal dari suatu file. Yang perlu diingat adalah *cluster* yang paling awal diakses adalah *cluster* #2, yang merupakan sebuah *root directory*, dimana direktori juga merupakan sebuah file. Terdapat empat tipe dari direktori 32 byte, yakni:

1. ***Normal record with short filename*** - Attrib normal
2. ***Long filename text*** – dengan attrib. keempat bit di set
3. ***Unused*** – byte pertamanya adalah 0xE5
4. ***End of directory*** – byte pertamanya adalah kosong

Agar data dapat diakses dengan mudah, penulisan sebuah direktori oleh microsoft memiliki aturan yang terlampir dalam Tabel 2.17 dan Tabel 2.18.

Tabel 2.17 Struktur short directory FAT 32

Name	Offset (byte)	Size (byte)	Description	
DIR_Name	0	11	Short name. (8 main letters and 3 for the extension)	
DIR_Attr	11	1	File attributes:	
			ATTR_READ_ONLY	0x01
			ATTR_HIDDEN	0x02
			ATTR_SYSTEM	0x04
			ATTR_VOLUME_ID	0x08
			ATTR_DIRECTORY	0x10
			ATTR_ARCHIVE	0x20
			ATTR_LONG_NAME	ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID
			The upper two bits of the attribute byte are reserved and should always be set to 0 when a file is created and never modified or looked at after that.	
DIR_NTRes	12	1	Reserved for use by Windows NT. Set value to 0 when a file is created and never modify or look at it after that.	
DIR_CrtTimeTenth	13	1	Millisecond stamp at file creation time. This field actually contains a count of tenths of a second. The granularity of the seconds part of DIR_CrtTime is 2 seconds so this field is a count of tenths of a second and its valid value range is 0-199 inclusive.	
DIR_CrtTime	14	2	Time file was created.	
DIR_CrtDate	16	2	Date file was created.	
DIR_LstAccDate	18	2	Last access date. Note that there is no last access time, only a date. This is the date of last read or write. In the case of a write, this should be set to the same date as DIR_WrtDate.	
DIR_FstClusHI	20	2	High word of this entry's first cluster number (always 0 for a FAT12 or FAT16 volume).	
DIR_WrtTime	22	2	Time of last write. Note that file creation is considered a write.	
DIR_WrtDate	24	2	Date of last write. Note that file creation is considered a write.	
DIR_FstClusLO	26	2	Low word of this entry's first cluster number.	
DIR_FileSize	28	4	32-bit DWORD holding this file's size in bytes.	

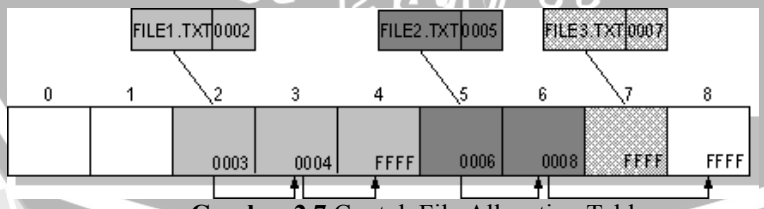
Sumber : Microsoft Corporation, 2000, *Microsoft Extensible Firmware Initiative FAT32 File System Specification FAT: General Overview of On-Disk Format*

Tabel 2.18 Struktur long directory FAT 32

Name	Offset (byte)	Size (bytes)	Description
LDIR_Ord	0	1	The order of this entry in the sequence of long dir entries associated with the short dir entry at the end of the long dir set. If masked with 0x40 (LAST_LONG_ENTRY), this indicates the entry is the last long dir entry in a set of long dir entries. All valid sets of long dir entries must begin with an entry having this mask.
LDIR_Name1	1	10	Characters 1-5 of the long-name sub-component in this dir entry.
LDIR_Attr	11	1	Attributes - must be ATTR_LONG_NAME
LDIR_Type	12	1	If zero, indicates a directory entry that is a sub-component of a long name. NOTE: Other values reserved for future extensions. Non-zero implies other dirent types.
LDIR_Chksum	13	1	Checksum of name in the short dir entry at the end of the long dir
			set.
LDIR_Name2	14	12	Characters 6-11 of the long-name sub-component in this dir
			entry.
LDIR_FstClusLO	26	2	Must be ZERO. This is an artifact of the FAT "first cluster" and must be zero for compatibility with existing disk utilities. It's meaningless in the context of a long dir entry.
LDIR_Name3	28	4	Characters 12-13 of the long-name sub-component in this dir entry.

Sumber : Microsoft Corporation, 2000, *Microsoft Extensible Firmware Initiative FAT32 File System Specification FAT: General Overview of On-Disk Format*

Dari direktori tersebutlah letak *cluster* awal dari data suatu file yang terpecah didapatkan. Kemudian pencarian letak cluster selanjutnya didapatkan dari pembacaan cluster awal tersebut, demikian seterusnya untuk pencarian cluster yang berikutnya hingga ditemukan cluster yang terakhir dari file tersebut, yang ditandai dengan cluster yang berisi heksa (0xFFF8–0xFFFF). Sedang untuk cluster yang rusak berisi heksa (0xFFF7), dan cluster yang kosong berisi heksa (0x0000 atau 0x00E5 -,yang merupakan hasil penghapusan suatu file,-). Sebagai penggambaran pencarian file terlihat pada Gambar 2.7.



Gambar 2.7 Contoh File Allocation Table
Sumber : NTFS, 2006, *File Allocation System*

2.2.2 USB (Universal Serial Bus)

Secara umum USB dirilis dengan tiga versi, pertama dengan versi 1.0 pada tahun 1996, kedua versi 1.1 pada tahun 1998 oleh Intel, dan ketiga versi

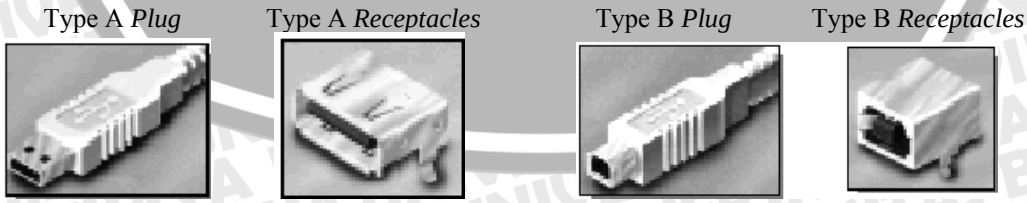
2.0 pada tahun 1999. USB versi 1.1 mempunyai kemampuan transfer data pada kecepatan rendah (*low speed*) dan kecepatan penuh (*full speed*), sedangkan pada versi 2.0 ditambah dengan kemampuan transfer data pada kecepatan tinggi (*high speed*). Adapun keterangan pada masing masing kecepatan tersebut adalah sebagai berikut:

- *High Speed* (480Mbits/s) dengan panjang data maksimum sebesar 1024 byte
- *Full Speed* (12Mbits/s) dengan panjang data maksimum sebesar 64 byte
- *Low Speed* (1.5Mbits/s) dengan panjang data maksimum sebesar 8 byte

Suatu piranti USB dapat dikatakan sebagai sebuah alat *transceiver* (*transmitter-receiver*), karena berperan sebagai pengirim sekaligus penerima. Karena piranti USB mengirim dan menerima data melalui beberapa *endpoint* -, sumber data ataupun titik tujuan data yang merupakan ujung saluran komunikasi pada USB,- yang terpasang seri, maka *software* disisi klien akan mentransfer data melalui *pipe*. *Pipe* adalah hubungan secara logika antara *host* dan *endpoint*. *Pipe* memiliki beberapa parameter antara lain berapa *bandwith* yang dialokasikan untuknya, jenis transfer apa yang digunakan (apakah *control*, *bulk*, atau *interupt*), serta arah dari data yang mengalir padanya berikut ukuran maksimum paket (atau ukuran *buffer*).

2.2.2.1 Konektor USB

Pada dasarnya terdapat dua tipe konektor USB, yakni konektor tipe A dan tipe B. Secara umum tipe A digunakan untuk hubungan ke atas (*host*), yakni cpu ataupun piranti *host* lainnya, dan konektor tipe B digunakan untuk hubungan ke bawah (piranti USB), yakni *printer*, *scanner* atau USB *device* lainnya. Bentuk fisik konektor tipe A dan B ditunjukkan dalam Gambar 2.8



Gambar 2.8. Bentuk konektor USB tipe A dan B
Sumber : Leong, Chui Wei, 2005, *Understanding Universal Serial Bus (USB)*, USBDeveloper

Untuk keperluan lebih luas seperti menghubungkan ke piranti berukuran kecil seperti PDA, handphone, dll., dibuatlah konektor mini-A dan mini-AB. Adapun pengkabelan pada USB tercantum pada Tabel 2.19

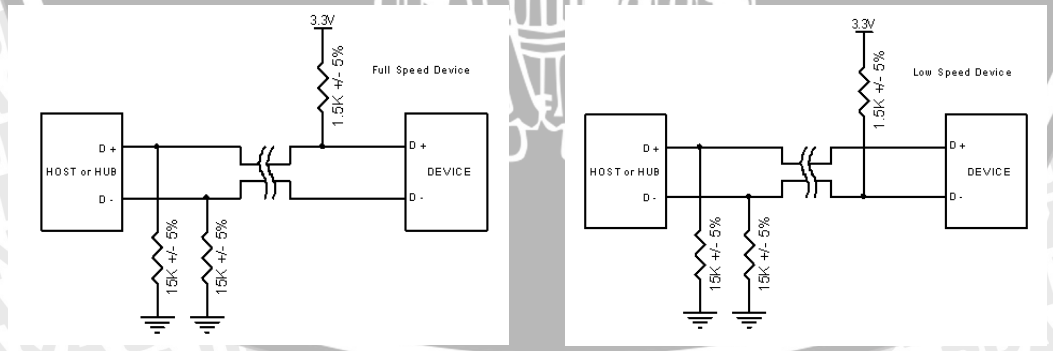
Tabel 2.19. Pengkabelan USB

Nomor pin	Fungsi	Warna kabel
1	VBus	*Red* (merah)
2	D-	*White* (putih)
3	D+	*Green* (hijau)
4	Gnd	*Black* (hitam)

Sumber : Leong, Chui Wei, 2005, *Understanding Universal Serial Bus (USB)*, USBDeveloper

2.2.2.2 Karakteristik elektris USB

Sebuah perangkat USB menunjukkan kecepatannya dengan menempatkan salah satu jalur D+ atau D- hingga sebesar 3,3V. Perangkat kecepatan penuh, menggunakan resistor *pull-up* pada jalur D+ untuk menunjukkan dirinya berkecepatan penuh. Sedang perangkat kecepatan rendah, menggunakan resistor *pull-up* pada jalur D– untuk menunjukkan dirinya berkecepatan rendah. Resistor *pull-up* pada ujung perangkat ini digunakan oleh *host* atau *hub* untuk mendeteksi keberadaan perangkat yang terhubung pada konektornya. Tanpa resistor *pull-up*, USB berasumsi bahwa tidak terdapat suatu apapun yang terhubung pada konektornya. Beberapa perangkat mempunyai resistor ini di dalam silikonnya, yang dapat diaktifkan dan dinonaktifkan di bawah kendali *firmware*, sedang yang lain membutuhkan resistor eksternal. Penempatan resistor *pull-up* diperlihatkan dalam Gambar 2.9



(a) (b)

Gambar 2.9. (a) Perangkat kecepatan penuh dengan resistor *pull-up* terhubung pada D+; (b) Perangkat kecepatan rendah dengan resistor *pull-up* terhubung pada D-; terminator *pull-down* di port *down-stream* adalah resistor 15kΩ ± 5% terhubung ke *ground*.

Sumber : Peacock, Craig, 2002,*USB in a Nutshell: Making Sense of the USB Standard, Beyond Logic*

Rentang tegangan kerja sinyal USB adalah 0,3V hingga 3,6V (pada beban 1,5kΩ), namun pada USB kecepatan tinggi rentang tegangan kerja sinyalnya antara 0,4V hingga 3,3V guna memastikan total daya yang ditransmisikan tetap rendah karena bekerja pada frekuensi 480Mbit/s. Logika tinggi didapat jika tegangan sudah melebihi 2,8V terhadap *ground* pada beban 15 kΩ. Pada piranti USB yang berkecepatan rendah dan penuh, differensial "1" dikirim dengan menarik D+ hingga lebih besar dari 2,8V dengan sebuah resistor 15kΩ terhubung dengan *ground* dan sekaligus menarik D- hingga di bawah 0,3V dengan sebuah resistor 1,5kΩ terhubung ke 3,6 volt. Hal yang sama, differensial "0" adalah D- lebih besar dari 2,8V dan D+ lebih rendah dari 0,3V dengan resistor *pull-up* dan *pull-down* yang sama. Di bagian penerima, differensial "1" didefinisikan sebagai D+ lebih besar 200mV dari D-, dan differensial "0" berarti D+ lebih kecil dari 200mV dibanding D-. Pada USB berkecepatan tinggi (480MBit/s) digunakan sumber arus tetap 7,78mA untuk mengurangi noise.

Pada USB polaritas sinyal berubah tergantung pada kecepatan dari BUS, oleh sebab itu istilah kondisi 'J' dan 'K' (*'J' and 'K' state*) digunakan untuk menandakan level logika. Pada kecepatan rendah kondisi 'J' diartikan sebagai deferensial 0, sedang pada kecepatan tinggi dan penuh kondisi 'J' diartikan sebagai deferensial 1. USB *transceiver* mempunyai kedua bentuk deferensial tersebut dan satu keluaran akhir (*single ended outputs*). Bentuk kondisi level logika untuk masing-masing kecepatan dijelaskan dalam Tabel 2.20

Tabel 2.20. Kondisi level logika pada USB kecepatan penuh, rendah dan tinggi

Kondisi	Kecepatan penuh		Kecepatan rendah		Kecepatan tinggi	
	D+	D-	D+	D-	D+	D-
J	High	Low	Low	High	High	Low
K	Low	High	High	Low	Low	High
Idle	High	Low	Low	High	Low	Low
SE0(*Single Ended Zero*)	Low	Low	Low	Low	-	-
SE1 (*Single Ended One*)	High	High	High	High	-	-
Chirp J state	-	-	-	-	High*	Low
Chirp K state	-	-	-	-	Low	High*

Sumber : Leong, Chui Wei, 2005, *Understanding Universal Serial Bus (USB)*, USBDeveloper

Pemastian kondisi bus ditandai dengan sebuah sinyal akhir pada D+, D-, atau keduanya. Sebagai contoh sebuah akhir nol (*single ended zero*) atau SE0 dapat digunakan untuk memberitahukan bahwa perangkat akan di reset jika

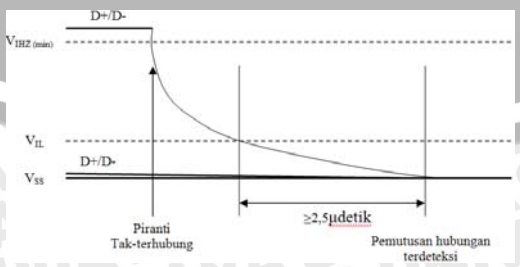
berhenti/tertahan selama lebih dari 10mS. SE0 dibangkitkan dengan menahan baik D+ atau D- dalam kondisi low (<0,3V).

Impedansi masukan di D+ dan D- tanpa terminal harus lebih dari 300kΩ. Sedang kapasitansi masukan dari port yang diukur pada pin konektor boleh berbeda pada port *upstream* dan *downstream*. Namun kapasitansi maksimum di *downstream* adalah 150pF pada D+ dan D-. Ini terdiri atas 75pF (maksimum) kapasitansi terhadap ground di *tranceiver* dan di konektor, ditambah tambahan 75pF kapasitansi di masing-masing konduktor di jalur transmisi antara pengirim dan penerima. Untuk *upstream*, kapasitansi maksimumnya adalah 100pF di D+ dan D-. Jadi tambahan kapasitornya hanya sebesar 25pF.

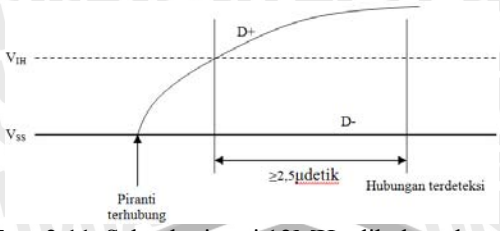
2.2.2.3 Sinyal penyambungan dan pemutusan

Ketika tidak ada piranti yang terhubung ke *host* atau *hub*, resistor *pull-down* (Gambar 2.9) yang ada menyebabkan D+ dan D- tertarik hingga di bawah ambang logika rendah *host* atau *hub*. Ini menyebabkan munculnya keadaan SE0 (*Single Ended Zero*) di *port downstream*. Kondisi tak terhubungnya suatu piranti USB dari *port* akan dideteksi jika *host* atau *hub* tidak men-*drive header* (di jalur data) selama lebih dari 2,5µdetik.

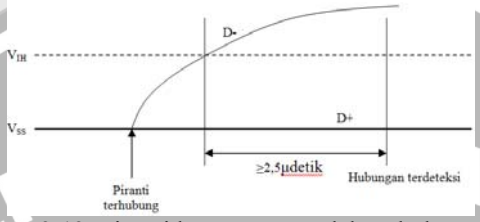
Jika sebuah piranti dihubungkan, maka *hub* akan mendeteksi bahwa salah satu jalur datanya tertarik hingga lebih besar dari ambang V_{IH} selama lebih dari 2,5 µdetik. *Hub* kemudian bisa (*option*) mendeteksi kecepatan piranti yang baru terhubung ini dengan mencuplik keadaan bus segera sebelum men-*drive* SE0 untuk mereset piranti. Jika diinginkan, *hub* dapat membuat bus mengambang sesudah meresetnya dan menjalankan evaluasi bus sesudah 2,5 µdetik tersebut, seperti yang terlihat dalam Gambar 2.10, 2.11, dan 2.12



Gambar 2.10. Sebuah piranti dilepas dari port
Sumber : Sutadi, Dwi. 2004. *I/O BUS & Motherboard*. ANDI. Yogyakarta



Gambar 2.11. Sebuah piranti 12MHz dihubungkan ke port USB
Sumber : Sutadi, Dwi. 2004. *I/O BUS & Motherboard*. ANDI. Yogyakarta



Gambar 2.12. Piranti kecepatan rendah terhubung ke port USB
Sumber : Sutadi, Dwi. 2004. *I/O BUS & Motherboard*. ANDI. Yogyakarta

Untuk mengirimkan paket data, USB menerapkan enkode data NRZI (*Non Return to Zero Invert*). Dalam NRZI ini, logika "1" berarti tidak ada perubahan level tegangan dan logika "0" ditunjukkan dengan adanya perubahan level tegangan. Beberapa literatur menyebutnya dengan NRZS (S=*Space*).

Paket data dikirimkan ke USB berurutan dari bit yang berbobot paling rendah (LSB, *least Significant Bit*), hingga yang terakhir adalah bit yang berbobot paling tinggi (MSB, *Most Significant Bit*).

2.2.3 Perintah pemonitoran VNC1L

Semua pemrosesan sistem USB dan sistem FAT, dapat dilakukan secara internal di dalam IC vinculum ini. Dimana untuk keperluan pengaksesan data dapat dilakukan melalui sebuah mikrokontroller dengan memberikan perintah yang tersedia dalam firmware VDAP, yang telah disediakan oleh vinculum, dengan menggunakan jalur komunikasi UART, SPI, atau parallel FIFO. Terdapat dua jenis perintah VNC1L–1A yakni, *Extended* dan *Short* (menggunakan kode Heksadesimal) seperti yang tercantum pada Tabel 2.21

Tabel 2.21. Perintah untuk memonitor vinculum

Extended Command Set	*Short Command Set*	Fungsi
SCS↵	10 0D	Beralih ke mode *shortened command set*
ECS↵	11 0D	Beralih ke mode *extended command set*

Tabel 2.21. Perintah untuk memonitor vinculum (lanjutan)

Extended Command Set	*Short Command Set*	Fungsi
IPA↵	90 0D	Perintah pemonitoran menggunakan nilai ASCII
IPH↵	91 0D	Perintah pemonitoran menggunakan nilai biner
SBD.divisor↵	14 20 divisor 0D	Ganti *baud rate*
FWV↵	13 0D	Mengetahui versi *firmware*
E↵	45 0D	Echo 'E' untuk sinkronisasi
E↵	65 0D	Echo 'e' untuk sinkronisasi
DIR↵	01 0D	Mendata file pada direktori yang sedang dibuka
DIR␣file↵	01 20 file 0D	Mendata file yang dimaksudkan dan ukurannya
CD␣file↵	02 20 file 0D	Ubah direktori yang sedang dibuka
CD␣..↵	02 20 2E 2E 0D	Menaikkan satu level Direktori
RD␣file↵	04 20 file 0D	Baca keseluruhan file
DLD␣file↵	05 20 file 0D	Menghapus subdirektori dari direktori yang sedang dibuka
MKD␣file↵	06 20 file 0D	Buat subdirektori baru dalam direktori yang sedang dibuka
MKD␣file␣datetime↵	06 20 file 0D	Buat subdirektori baru dalam direktori yang sedang dibuka dengan menyertai tanggal dan waktu pembuatan
DLF␣file↵	07 20 file 0D	Menghapus sebuah file
WRF␣dword↵data	08 20 dword 0D data	Menuliskan sejumlah byte yang diinginkan mulai dari parameter pertama dari file yang sedang dibuka
OPW␣file↵	09 20 file 0D	Membuka sebuah file untuk ditulis atau membuat file baru
OPW␣file␣datetime↵	09 20 file 20 datetime 0D	Membuka sebuah file untuk ditulis atau membuat file baru disertai tanggal dan waktunya
CLF␣file↵	0A 20 file 0D	Menutup file yang sedang dibuka
RDF␣dword↵	0B 20 dword 0D	Membaca sejumlah byte yang diinginkan mulai dari parameter pertama dari file yang sedang dibuka
REN␣file␣file↵	0C 20 file 20 file 0D	Penamaan ulang sebuah file atau direktori
OPR␣file↵	0E 20 file 0D	Membuka sebuah file untuk dibaca
OPR␣file␣date↵	0E 20 file 20 date 0D	Membuka sebuah file untuk dibaca disertai tanggal pengaksesan
SEK␣dword↵	28 20 dword 0D	Mencari byte untuk dijadikan sebagai parameter pertama dari file yang sedang dibuka
FS↵	12 0D	Mengetahui ruang kosong yang tersedia pada disk jika kurang dari 4GB
FSE↵	93 0D	Mengetahui ruang kosong yang tersedia pada disk
IDD↵	0F 0D	Mengetahui informasi tentang disk jika kurang dari 4GB
IDDE↵	94 0D	Mengetahui informasi tentang disk
DSN↵	2D 0D	Mengetahui nomor serial dari disk
DVL↵	2E 0D	Mengetahui label volume dari disk
DIRT␣file↵	2F 20 file 0D	Menjabarkan tanggal dan waktu pembuatan, pengubahan dan pengaksesan terakhir file

Tabel 2.21. Perintah untuk memonitor vinculum (lanjutan)

Extended Command Set	*Short Command Set*	Fungsi
SUD↵	15 0D	Mengistirahatkan disk
WKD↵	16 0D	Membangunkan disk
SUM↵	17 0D	Mengistirahatkan monitor
IOR␣byte↵	29 20 byte 0D	Membaca Port I/O (parameter pertama merupakan nomor port)
IOW␣byte+byte+byte↵	2A 20 byte byte byte 0D	Menulis Port I/O (parameter pertama merupakan nomor port, kedua adalah arahnya, ketiga adalah nilainya)
PGS↵	81 0D	Mengetahui status printer
PSR↵	82 0D	*Printer soft reset*
QP1↵	2B 0D	*Query port 1*
QP2↵	2C 0D	*Query port 2*
QD␣byte↵	85 20 byte 0D	*Query device* dijabarkan dalam parameter pertama
SC␣byte↵	86 20 byte 0D	Menyetel perangakat yang dijabarkan pada parameter pertama sebagai perangkat yang sedang digunakan
DSD␣byte↵data	83 20 byte 0D data	Mengirim data ke perangkat USB dimana ukuran data dijabarkan pada parameter pertama
DRD↵	84 0D	Membaca kembali data dari perangkat USB
SSU␣qword↵data	9A 20 qword 0D	Mengirim data setup ke endpoint control dari suatu perangkat
SF ␣ byte↵	87 20 byte 0D	Menyetel perangkat yang dijabarkan dalam parameter pertama sebagai perangkat FTDI
QSS↵	98 0D	*Query Slave Status* (hanya bekerja pada VDPS)
FBD ␣ divisor↵	18 20 divisor 0D	Menyetel *baut rate*
FMC ␣ word↵	19 20 word 0D	Menyetel control modem
FSD ␣ word↵	1A 20 word 0D	Menyetel karakteristik data
FFC ␣ byte↵	1B 20 byte 0D	Menyetel kontrol aliran
FGM↵	1C 0D	Mengetahui status modem
FSL␣byte↵	22 20 byte 0D	Mengatur pewaktuan
FSB␣word↵	23 20 word 0D	Menyetel mode bit
FGB↵	24 0D	Mengetahui mode bit
VPF␣file↵	1D 20 file 0D	Memainkan satu buah file
VST↵	20 0D	Menghentikan *playback*
V3A↵	21 0D	Memainkan semua file MP3
VSF↵	25 0D	Lewati satu *track* selanjutnya
VSB↵	26 0D	Lewati satu *track* sebelumnya
VRD␣byte↵	1F 20 byte 0D	Membaca register perintah
VWR␣byte+word↵	1E 20 byte word 0D	Menulis register perintah

Tabel 2.21. Perintah untuk memonitor vinculum (lanjutan)

Extended Command Set	*Short Command Set*	Fungsi
VSV␣byte↵	88 20 byte 0D	Menyetel volume *playback*
Playing␣file↵	P␣file↵(50 20 file 0D)	Memulai *track*
Stopped↵	S↵(52 0D)	Playback dihentikan
SD␣dword↵	03 20 dword 0D	*Sector dump*
SW␣dword↵data	92 20 dword 0D data	*Sector write*
FWU␣file↵	95 20 file 0D	Meng-*upgrade firmware* dari nama file yang disebutkan dalam disk

Sumber : Future Technology Devices International Ltd. (FTDI), 2007, *Future Technology Devices International Ltd Vinculum Firmware User Manual Version: 2.1 l*, Glasgow, UK

2.2.4 Pemrograman VNC1L

Agar dapat digunakan IC VNC1L-1A ini harus terlebih dahulu diisi dengan salah satu program yang telah disediakan oleh FTDI dalam situsnya http://www.vinculum.com/downloads.html, antara lain:

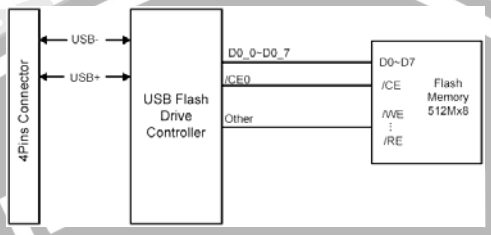
- VDAP *Disk And Peripheral Firmware*
- VMSC *Music Firmware*
- VDPS *Disk or Peripheral Firmware*
- VCDC *Communication Device Class Firmware*
- VDIF *Disk Interface Firmware*
- VDFC *Disk File Copy Firmware*
- VF2F *Firmware*

Firmware yang disediakan tersebut di-*upload*-kan ke dalam IC VNC1L–1A dengan perantara IC FT232BM/FT232BL menggunakan software "VPROG" yang telah disediakan dalam situs vinculum. Sebelum meng–*upload* program, IC VNC1L–1A di-*set* terlebih dahulu ke mode komunikasi UART, serta menghubungkan pin10 (PROG#) ke ground. *Default* UART VNC1L–1A adalah baudrate sebesar 9600, 8 bit data, 1 bit *start*, 1 bit *stop*, dan tanpa paritas dengan sinyal handshaking RTS/CTS aktif.

Untuk merubah nilai default tersebut dapat dilakukan dengan mengaplikasikan software "VncFwMod" yang juga telah disediakan oleh vinculum dalam situs-nya atau melalui mode perintah (*command mode*).

2.3 USB *Flash Drive*

Flash Drive pada dasarnya merupakan gabungan dari memori, USB driver, dan USB *port*. Diagram blok sistem flash *drive* dapat dilihat dalam Gambar 12 di bawah, penggambaran *Flash Drive* ini merupakan contoh dari flash *drive* tipe TS512MJF110 buatan Transcend Inc.

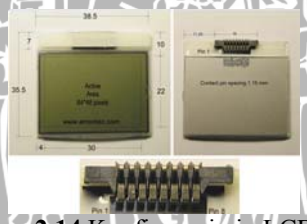


Gambar 2.13. Blok Diagram *Flash Drive*
Sumber : Transcend Information Inc., 2006, *TS512MJF110: 512MB USB2.0 JetFlash™*

2.4 LCD (Liquid Cristal Display) Nokia 3310

LCD Nokia 3310 yang diproduksi oleh Nokia memiliki spesifikasi:

a.) Ukuran display yakni 38x35 mm, lebar display aktif sebesar 30x22mm.
b.) Resolusi sebesar 84x48 pixel
c.) Menggunakan komunikasi SPI
d.) Rentang tegangan antara Vdd terhadap Vss sebesar 2.7 hingga 3.3 V.
e.) Konsumsi daya rendah, cocok untuk sistem yang menggunakan baterai.



Gambar 2.14 Konfigurasi pin LCD Nokia 3310
Sumber : Amontec. 2004 , *Nokia 3310 LCD 84x48 pixels*

Fungsi tiap-tiap pin LCD Nokia 3310 dijabarkan dalam Tabel 2.22

Tabel 2.22. Konfigurasi pin LCD 3310

Pin	Signal	Deskripsi	*Port*
1	VDD	Rentang tegangan terhadap ground (2.7 – 3.3V)	Catu daya
2	SCLK	Pewaktu serial (0.0 – 4.0 Mbit/s)	Masukan
3	SDIN	Jalur masukan data serial	Masukan
4	D/C#	Pemilih mode masukan (mode perintah(0)/data(1))	Masukan
5	SCE	Masukan *chip enable* (aktif *low*)	Masukan
6	GND	*Ground*	Catu daya
7	Vout	Tegangan keluaran (dihubungkan dengan *ground* dengan perantara kapasitor 1 - 10µF)	Catu daya
8	RES	Reset (aktif *low*)	Masukan

Sumber : Amontec. 2004

LCD Nokia 3310 memiliki driver internal yang mengontrol penulisan dalam LCD ini, yakni PCD8544, yang memiliki sepesifikasi sebagai berikut :

a.) *Single chip* LCD *controller*

b.) 48 baris X 84 kolom (Display data RAM 48 x 84 bit)

Perintah LCD nokia 3310 dapat dilihat dalam Tabel 2.23.

Tabel 2.23.Perintah untuk LCD Nokia 3310

Instruksi	D/C#	Byte perintah								Deskripsi
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 atau 1)										
NOP	0	0	0	0	0	0	0	0	0	Tidak ada operasi
Set fungsi	0	0	0	0	1	0	PD	V	H	Power down control (PD), Mode entry (V) Set instruksi tambahan (H)
Tulis data	1	D7	D6	D5	D4	D3	D2	D1	D0	Tulis data RAM tampilan
(H = 0)										
Reserved	0	0	0	0	0	0	1	X	X	Tidak digunakan
Kontrol layer	0	0	0	0	0	1	D	0	E	Set konfigurasi tampilan
Reserved	0	0	0	0	1	X	X	X	X	Tidak digunakan
Set alamat Y RAM	0	0	1	0	0	0	Y2	Y1	Y0	$0 \leq Y \leq 5$
Set alamat X RAM	0	1	X6	X5	X4	X3	X2	X1	X0	$0 \leq X \leq 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	Tidak digunakan
Reserved	0	0	0	0	0	0	0	1	X	Tidak digunakan
Kendali temperatur	0	0	0	0	0	0	1	TC1	TC0	Set koefisien temperature (TCx)
Reserved	0	0	0	0	0	1	X	X	X	Tidak digunakan
System Bias	0	0	0	0	1	0	BS2	BS1	BS0	Set system Bias (Bsx)
Reserved	0	0	1	X	X	X	X	X	X	Tidak digunakan
Set VOP	0	1	VOP6	VOP5	VOP4	VOP3	VOP2	VOP1	VOP0	Tulis VOP ke register

Sumber : Philips Semiconductors, 1999, *DATA SHEET: PCD8544 48 ´ 84 pixels matrix LCD controller/driver*, EINDHOVEN, The Netherlands,

Tabel 2.24. Keterangan symbol pada Tabel 2.29

Bit		0	1
PD		*Chip* aktif	*Chip* dalam mode *Power down*
V		Pengalamatan horisontal	Pengalamatan vertikal
H		Set instruksi dasar	Set instruksi tambahan
D	**E**		
0	0	*Display blank*	
1	0	*Normal mode*	
0	1	Semua segmen layer menyala	
1	1	*Inverse video mode*	
TC1	**TC0**		
0	0	Koefisien temperatur VLCD 0	
1	0	Koefisien temperatur VLCD 1	
0	1	Koefisien temperatur VLCD 2	
1	1	Koefisien temperatur VLCD 3	

Sumber : Philips Semiconductors, 1999, *DATA SHEET: PCD8544 48 ´ 84 pixels matrix LCD controller/driver*, EINDHOVEN, The Netherlands,

Driver PCD8544 yang telah terintegrasi dalam LCD Nokia 3310 ini, selain menuliskan sebuah bit dalam satu pixel LCD, juga mampu mengontrol pada pointer mana karakter mulai dituliskan.

Setiap pengiriman data pada mode data disalin menjadi tampilan 8 buah pixel dengan arah vertikal oleh driver PCD8544, dikarenakan komunikasi yang digunakan adalah komunikasi SPI. Dengan demikian karena LCD berukuran 48 × 84 pixel (lebar = 84 pixel; panjang = 48 pixel), maka pointer bergerak dengan sumbu dua dimensi yakni horisontal (X) dari X(0) hingga X(83) dan vertikal (Y) dari Y(0) hingga Y(5). Penempatan pointer ini tidak perlu dilakukan tiap kali mengirimkan sebuah data dari sejumlah paket data, cukup menentukan poiter awal dimana data akan ditulis kemudian PCD menambahkan pointer secara otomatis hingga data yang terakhir.

Pola pengalamatan tampilan di LCD ini terdapat dua pola yakni pola pengalamatan vertikal dan kedua pola pengalamatan horisontal. Pola pengalamatan vertikal yakni sejumlah data yang dikirim pertama-tama akan dituliskan secara vertikal hingga berada pada pointer X(n),Y(5), dimana n adalah sembarang data dari 0 hingga 83, kemudian secara otomatis digeser secara horisontal menjadi pointer X(n+1),Y(0), bila (n) adalah 83 maka kemudian (n+1) adalah 0, demikian seterusnya hingga data yang terakhir.

Pola pengalamatan horisontal yakni sejumlah data yang dikirim pertama-tama akan dituliskan secara horisontal hingga berada pada pointer X(83),Y(n), dimana n adalah sembarang data dari 0 hingga 5, kemudian secara otomatis digeser secara vertikal menjadi pointer X(0),Y(n+1), bila (n) adalah 5 maka kemudian (n+1) adalah 0, demikian seterusnya hingga data yang terakhir.

LCD nokia 3310 memiliki RAM internal yang berfungsi untuk menyimpan sementara sejumlah paket data yang ditampilkan hingga catu daya dilepas.

LCD ini tidak mempunyai database data layaknya LCD 16 × 2 dan sejenisnya, oleh sebab itu karakter atau tampilan yang ingin ditampilkan perlu disiapkan terlebih dahulu dan diubah menjadi sejumlah paket data yang

dikirimkan ke LCD nokia3310. Salah satu *software* yang mampu menterjemahkan rancangan tampilan LCD menjadi sejumlah data matang yang siap dikirimkan ke LCD nokia3310 adalah ”fastLCD” buatan Bojan I.

BAB III
METODOLOGI

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat menampilkan unjuk kerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Data dan spesifikasi komponen yang digunakan dalam perencanaan merupakan data sekunder yang diambil dari buku data komponen elektronika.

Langkah-langkah yang perlu dilakukan untuk merealisasikan alat yang akan dibuat adalah sebagai berikut:

3.1. Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perencanaan dan perealisasian alat. Adapun teori-teori yang dikaji adalah sebagai berikut:

1.) SPI
2.) LCD Nokia 3310
3.) USART
4.) Vinculum VNC1L-1A
5.) USB *Flash Drive*

3.2. Perancangan Alat

Berdasar studi literatur yang diteruskan pada tahap selanjutnya yaitu perancangan alat. Hal ini berhubungan dengan perancangan instrumen alat pengkopi data antar *flash drive* yang meliputi penyusunan blok diagram sistem, pembuatan skema rangkaian, penentuan dan perhitungan komponen yang akan digunakan. Perancangan rangkaian ini dilakukan untuk masing-masing blok yaitu :

1.) Blok LCD Nokia 3310
2.) Blok VNC1L
3.) Blok Rangkaian ATmega64

3.3. Pembuatan Alat

Pembuatan alat dilakukan didasarkan dari perencanaan yang telah dibuat, dimulai dengan pembuatan unit rangkaian tiap blok pada *project board* yang selanjutnya dilakukan pengujian awal. Kemudian tiap blok-blok rangkaian pada project board digabungkan. Setelah dirangkai tiap-tiap blok sesuai dengan perencanaan maka dilakukan pembuatan PCB untuk tiap rangkaian per blok. Langkah awal pembuatan PCB dilakukan perancangan tata letak komponen dengan menggunakan *software eagle*. Kemudian *layout* PCB disablon dan dietsakan untuk melarutkan tembaga. Setelah proses pengetsaan selesai zat penutup sketsa rangkaian dibersihkan dengan alkohol selanjutnya dilakukan pengeboran lubang tempat kaki-kaki komponen. Langkah selanjutnya adalah penyolderan komponen dan perakitan menjadi satu rangkaian.

3.4. Pengujian Alat

Untuk mengetahui unjuk kerja piranti apakah sesuai dengan yang direncanakan maka dilakukan pengujian rangkaian. Pengujian dilakukan pada masing-masing blok dan secara keseluruhan. Pengujian antara lain berupa:

3.4.1. Pengujian Blok LCD Nokia 3310

Pengujian ini dilakukan dengan menyusun diagram alir perangkat lunak mikrokontroller untuk menangani penampil huruf/karakter ke dalam LCD Nokia 3310, kemudian dilanjutkan dengan merancang program sesuai dengan diagram alir tersebut. Setelah program dimasukkan ke dalam mikrokontroller, selanjutnya menghubungkan jalur komunikasi SPI antara minkrokontroller dengan LCD Nokia 3310 dan melihat hasil tampilan pada LCD tersebut.

Tujuan dari pengujian ini adalah untuk mengetahui apakah LCD dapat menampilkan karakter yang ingin ditampilkan dengan baik atau tidak.

3.4.2. Pengujian Blok Vinculum VDIP2

Pengujian rangkaian ini dilakukan dengan merangkai rangkaian vinculum F2F, kemudian dilanjutkan dengan memasukkan firmware VDAPFUL_V3_56

yang disediakan oleh vinculum ke dalam IC VNC1L-1A. Setelah firmware dimasukkan ke dalam VNC1L-1A, dilanjutkan dengan menancapkan *Flash drive* sumber dan target -, yang mana *Flash drive* target dalam keadaan kosong,- kedalam Vinculum, dan melakukan proses pengkopian, kemudian melihat hasil pengkopian tersebut melalui komputer dengan menancapkan *flash drive* target ke dalam komputer.

Tujuan dari pengujian ini adalah untuk mengetahui apakah blok Vinculum DIP2 tersebut bekerja dengan baik apa tidak.

3.4.3. Pengujian Keseluruhan Sistem

Pengujian terhadap sistem pengkabelan dan hubungan tiap komponen dengan multimeter untuk mengetahui kesalahan penghubungan, peletakan dan terjadinya hubung singkat. Kemudian dilanjutkan dengan menghubungkan masing-masing blok menjadi satu. Pengujian ini juga dilakukan untuk mengetahui unjuk kerja alat telah bekerja dengan baik atau tidak.

3.5. Pengambilan Kesimpulan

Dari pengujian yang dilakukan diperoleh hasil yang dapat dianalisis. Hasil analisis tersebut akan dijadikan dasar untuk menarik kesimpulan dari proses perancangan sistem ini.

BAB IV
PERANCANGAN dan PEMBUATAN ALAT

Bab ini membahas tentang perancangaan dan pembuatan Alat Pengkopi data antar *flash drive* menggunakan ATmega64 dengan antar muka VNC1L yang meliputi perancangan perangkat keras dan perangkat lunak. Perancangan perangkat keras terdiri dari: rangkaian antarmuka LCD 3310 dengan sistem mikrokontroler Atmega64, rangkaian blok VNC1L yang meliputi rangkaian antar muka USB dengan vinculum dan rangkaian sistem vinculum VDIP versi 2, sistem mikrokontroller Atmega64 serta periperal pendukung yang berupa tombol 3×2 dan sebuah LCD Nokia 3310. Sedangkan perancangan perangkat lunak akan dibahas tentang pembuatan *flowchart* dan program *assembly* ATmega64.

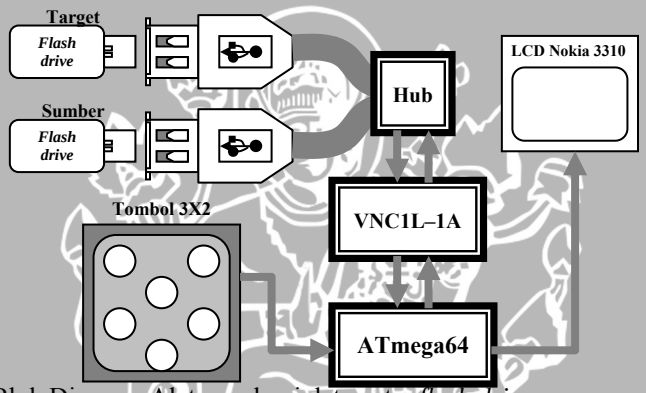
4.1. Spesifikasi Alat

Hal pertama yang perlu dilakukan sebelum melakukan perancangan dan pembuatan alat adalah penentuan spesifikasi alat yang akan dibuat. Spesifikasi alat yang akan direncanakan adalah sebagai berikut:

1.) Sumber tegangan berupa sumber tegangan DC antara 6 – 7.2 V. yang didapat dari 5 – 6 buah baterai sanyo AA 2700 mAh *rechargeable*.
2.) Sebuah *recharger internal* yang langsung dihubungkan dengan sumber tegangan AC 220 V.
3.) Sebuah IC Vinculum VNC1L-1A yang berfungsi untuk menangani protokol USB dan sistem file FAT.
4.) Sebuah LCD Nokia 3310 yang berfungsi untuk menampilkan menu pemilihan object yang akan dikopi serta peletakan target pengkopian.
5.) Sebuah mikrokontroller ATmega64 yang berfungsi sebagai pengendali utama sistem "pengkopi data antar *flash drive*"
6.) Kendali sistem percabangan USB Hub yang dilakukan oleh mikrokontroller ATmega64 dengan mengendalikan 2 buah jalur catu daya USB melalui sebuah relay.

4.2. Gambaran Umum

Alat pengkopi data antar flashdisk ini adalah alat portable yang digunakan untuk melakukan proses pentransferan data, dalam hal ini dipusatkan pada masalah pengkopian data antar flash disk melalui kendali keypad. Pembuatan alat ini ditujukan untuk membantu proses pengkopian data antar flash disk agar dapat berjalan lebih efektif dan efisien. Dengan alat ini pengguna tidak perlu lagi disibukkan dengan mencari sebuah komputer untuk melakukan proses pengkopian data. Dengan menggunakan alat ini pengguna dapat melakukan proses pengkopian dimana saja dan kapan saja.



Gambar 4.1. Blok Diagram Alat pengkopi data antar *flash drive* menggunakan Atmega64 dengan antar-muka VNC1L–1A

Pengkopian data dimulai dengan pencarian sumber data yang akan dikopi dengan cara menekan tombol pada *keypad* dan menampilkan hasil pencarian tersebut ke dalam LCD. Setelah data yang akan dikopi di dapat maka, mikrokontroller mengunci alamat tempat data tersebut berada pada flashdisk ke dalam EEPROM internal. Proses kemudian dilanjutkan dengan memilih tempat yang menjadi sasaran pengkopian dengan cara yang sama dengan proses pencarian sumber data yang akan dikopi. Setelah semuanya selesai maka proses pengkopian dilakukan dengan menekan tombol *copy* pada keypad.

4.3. Perancangan Perangkat Keras

Pada bagian ini membahas uraian rinci dari alat pengkopi data antar *flash drive* menggunakan ATmega64 dengan antar muka VNC1L, yang meliputi:

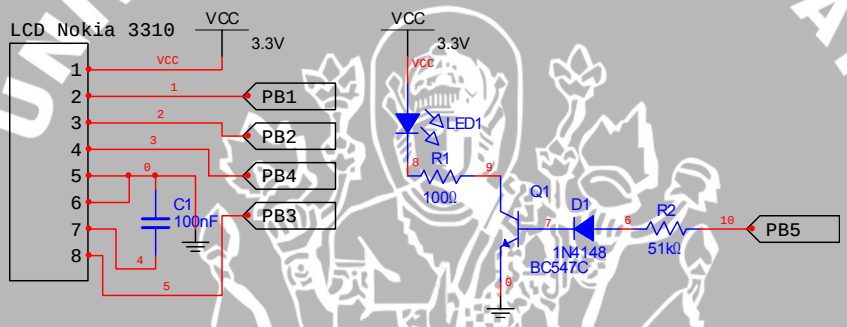
1.) Blok LCD Nokia 3310

2.) Blok VNC1L

3.) Blok Rangkaian ATmega64

4.3.1. Blok LCD Nokia 3310

Pada blok rangkaian ini hanya menambahkan beberapa rangkaian yakni sebuah resistor yang diantar-mukakan pada jalur *SCLK*, *SDIN*, *D* /C̄, dan *RESET* dengan pemasangan dioda zener 3,3V pada masing – masing jalur tersebut terhadap ground, serta sebuah rangkaian untuk *back light* LCD. Sedang pemberian kapasitor 100nF pada V_{out} digunakan sebagai kapasitor kopling yang mengopel tegangan keluaran dari LCD Nokia 3310.



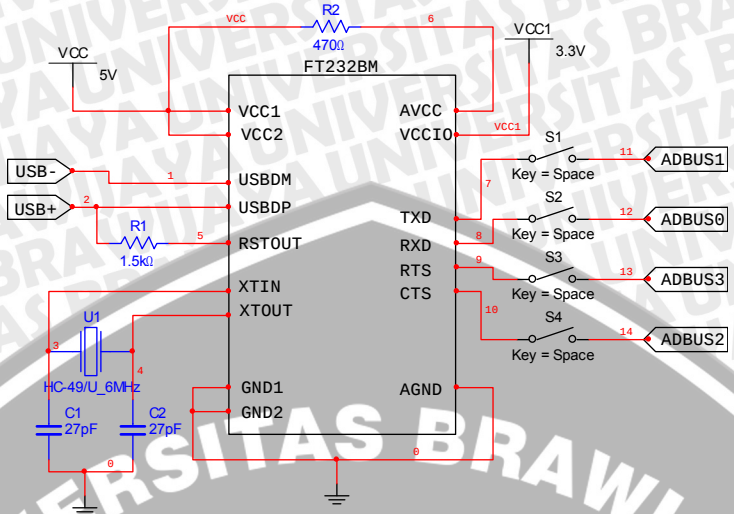
Gambar 4.2. Skematik rangkaian LCD Nokia 3310 + *back light* LCD

4.3.2. Blok VNC1L

Pada blok rangkaian ini terdapat dua buah sub blok rangkaian, yakni rangkaian *boot-loader* VNC1L-1A dan rangkaian VNC1L-1A, dimana rangkaian keduanya dihubungkan dengan tujuh buah *switch* yang berfungsi untuk mengatur jalur catu daya dan jalur bus data. Empat buah switch mengatur jalur transmisi UART, yakni RX, TX, RTS# dan CTS#. Tiga buah switch mengatur jalur catu daya yang diambil dari USB konektor pada komputer.

4.3.2.1. Rangkaian Boot-loader VNC1L-1A

Pada rangkaian ini device tidak berfungsi sebagai USB *Host* melainkan sebagai USB *Device* mode *Full–speed*. Sehingga untuk pengantar-mukaan dengan USB Hub pada komputer perlu disesuaikan dengan spesifikasi USB 2.0 untuk USB *Device*.



Gambar 4.3. Skematik rangkaian *Boot-loader* VNC1L-1A

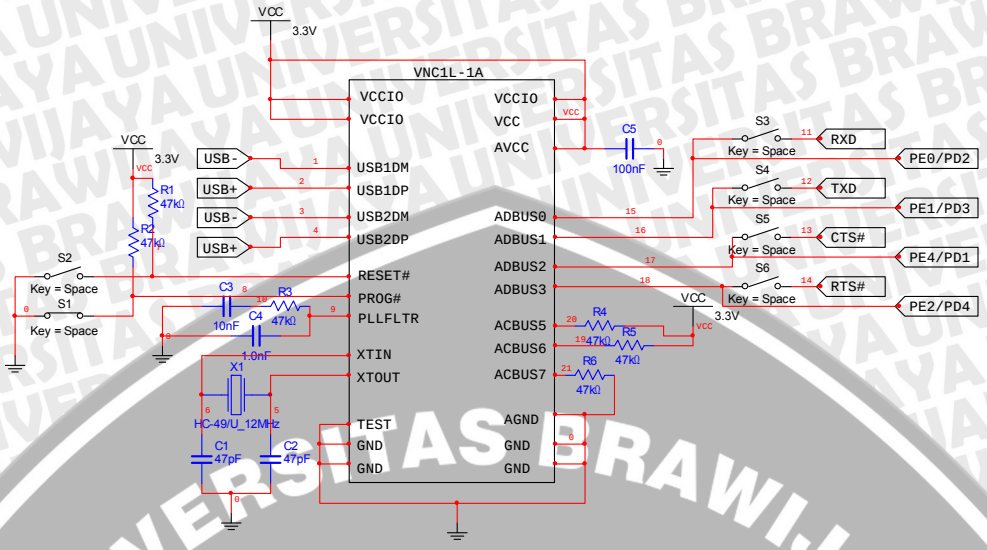
Pada rangkaian ini sumber tegangan diambil dari pin tegangan USB konektor pada komputer, yakni sebesar 5 V yang kemudian disesuaikan menjadi 3.3 V dengan menggunakan IC regulator AIC1722. Sumber tegangan 5 V digunakan untuk mencatu IC FT232BM untuk dapat bekerja, sedang sumber tegangan 3.3 V digunakan unruk mencatu jalur transmisi data UART. Sehingga level tegangan jalur transmisi UART menjadi :

- ***VOH*** **(**Output Voltage High) = 2.7 - 3.2 V dengan I source = 2mA
- ***VOL*** **(**Output Voltage Low) = 0.1 - 0.7 V dengan I sink = 4 mA.

Banyaknya pemasangan header di pin yang tidak digunakan sebagai jalur transmisi data dengan Vinculum ditujukan untuk pemakaian FT232BM secara umum.

4.3.2.2. Rangkaian VNC1L-1A

Rangkaian ini merupakan rangkaian kombinasi antar rangkaian VDIP2 dengan rangkaian VF2F. Dimana penyetelan model rangkaian dilakukan dengan menyetel jumper pada jumper yang tersedia. Sedang penentuan model komunikasi dengan Atmega64 dilakukan dengan menyetel jumper yang terhubung pada pin 46 (ACBUS5) dan 47 (ACBUS6) IC VNC1L–1A.



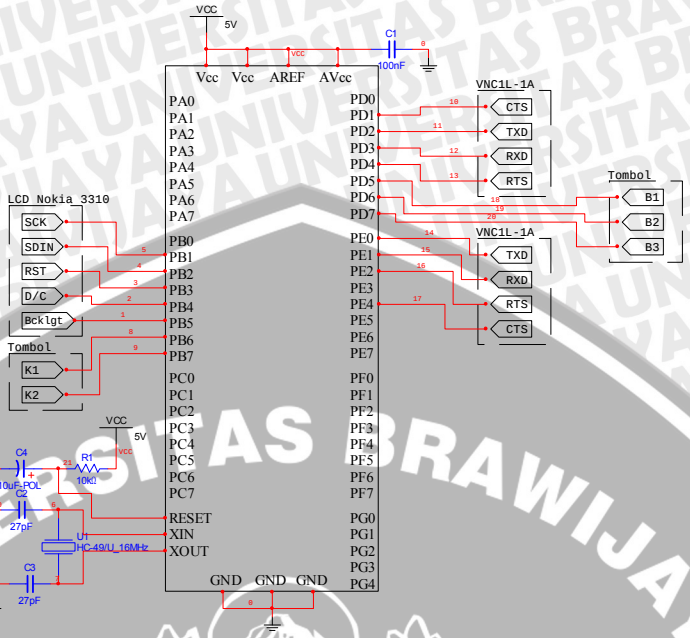
Gambar 4.4. Skematik rangkaian VNC1L-1A

IC ini mampu berperan sebagai USB *host*/*slave* versi 2.0 dengan mode kecepatan rendah/penuh. Sehingga untuk antar–muka perlu disesuaikan dengan spesifikasi USB host 2.0.

Sumber tegangan dari rangkaian ini diambil dari dua sumber yang berbeda. Sumber pertama didapatkan pada saat mem-*flash*-kan program ke dalam VNC1L–1A melalui IC FT232BM. Dimana sumber tegangan tersebut didapatkan dari tegangan keluaran USB port (5 V) dan IC regulator AIC1722 (3.3 V). Sumber kedua didapatkan dari catu daya eksternal yang berasal dari baterai. Pemilihan sumber tegangan ini dilakukan dengan menyetel *switch power*.

4.3.3. Blok Rangkaian ATmega64

Rangkaian ATmega64 ini merupakan pusat dari kendali semua sistem "Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega64 dengan Antar-muka VNC1L". Rangkaian ini berfungsi untuk mengirimkan perintah yang harus dijalankan oleh IC VNC1L–1A serta mengirimkan perintah untuk memberikan tampilan dalam LCD Nokia 3310.



Gambar 4.5. Skematik blok ATmega64

Mikrokontroler Atmega64 memiliki tujuh buah port dengan 53 jalur I/O yang bersifat *bidirectional*. Masing-masing I/O telah dilengkapi dengan resistor *pull up* internal. Pada perancangan ini pin-pin yang digunakan adalah:

1.) Port B(1, 2, 3, 4 dan 5) : Jalur komunikasi SPI dengan LCD Nokia 3310

2.) Port B(6 dan 7) : Jalur kolom 1 dan 2 untuk fungsi tombol

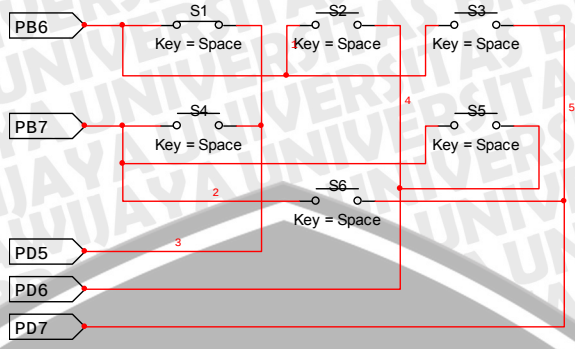
3.) Port D(1, 2, 3 dan 4) : Jalur komunikasi UART dengan VNC1L–1A

4.) Port D(5, 6 dan 7) : Jalur baris 1, 2 dan 3 untuk fungsi tombol

5.) Port E(0, 1, 2 dan 4) : Jalur komunikasi UART dengan VNC1L–1A

4.3.3.1. Tombol 2X3

Dalam perancangan ini tombol digunakan untuk memberikan input kepada mikrokontroler. Tombol yang digunakan adalah tombol 2X3. Proses pembacaan tombol dilakukan melalui proses *scanning* yaitu dengan cara memberikan logika 0 secara bergantian pada bagian baris kemudian mengecek logika 0 tersebut pada bagian kolom, proses ini dilakukan secara berulang-ulang dan dikendalikan oleh mikrokontroler.



Gambar 4.6. Tombol 2x3

Untuk dapat membedakan antar tombol, maka dilakukan pengkodean berdasarkan urutan baris dan kolom. Data pengkodean tombol – tombol dapat dilihat dalam Tabel 4.1.

Tabel. 4.1 Kode tombol

Saklar	Baris		Kolom		
	PB6	PB7	PD5	PD6	PD7
S1	√	-	√	-	-
S2	√	-	-	√	-
S3	√	-	-	-	√
S4	-	√	√	-	-
S5	-	√	-	√	-
S6	-	√	-	-	√

4.4. Perancangan Perangkat Lunak

4.4.1. Alokasi Memori

ATmega64 memiliki tiga buah memori yakni SRAM, EEPROM, dan Flash Memori. Pemakaian memori untuk rancangan ini dilakukan sebagai berikut:

- EEPROM ($0000 – $07FF) :

  EEPROM dialokasikan untuk menjejak letak file berada, mengetahui nomor urut file dalam satu listing direktori yang dikirim oleh Vinculum, mengetahui jumlah direktori yang dipakai untuk penjejak file berada, dan utnuk mengetahui jumlah file direktori sumber yang tidak digunakan sebagai penjejak oleh target. Alokasi pengalamatan EEPROM secara detail sebagai berikut:

  - Penjejak file sumber : 0x0000 – 0x03B7 (952 byte)

- Penjejak file target : 0x03B8 – 0x076F (952 byte)
- Perekam nomor urut file sumber : 0x0770 – 0x07B3 (68 byte)
- Perekam nomor urut file target : 0x07B4 – 0x07F7 (68 byte)
- Perekam jumlah urutan direktori sumber : 0x07F8 (1 byte)
- Perekam jumlah urutan direktori target : 0x07F9 (1byte)
- Perekam jumlah urutan direktori yang tidak terpakai : 0x07FA (1byte)

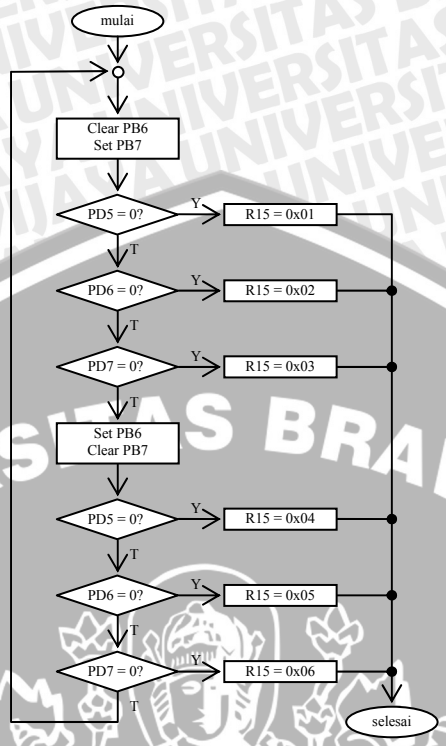
- SRAM ($0100 – $10FF)

  SRAM digunakan untuk menyimpan sementara data yang didapatkan dari Vinculum dengan alokasi maksimum se 3 Kbyte.

  Proses pengkopian ini dilakukan pertama-tama melakukan pembacaan direktori pada *flash drive* kemudian menyimpannya ke dalam SRAM, dilanjutkan dengan pengkonversian data untuk ditampilkan ke dalam LCD Nokia 3310. Saat sebuah file atau direktori dipilih, nama file beserta jenis ekstensinya (jenis ekstensi DIR atau bukan) dikunci dalam EEPROM dengan alamat dasar $0000 untuk *flash drive* sumber dan alamat dasar $03B8 untuk *flash drive* target. Alamat dasar tersebut akan bertambah secara otomatis seiring dengan bertambahnya file direktori yang dibuka, dengan batas maksimum alamat $03B7 untuk *flash drive* sumber dan $076F untuk *flash drive* target. Proses ini berlangsung hingga letak sumber dan tujuan pengkopian diketemukan. Setelah sumber diketemukan maka data yang akan dikopi diambil sebanyak 3 Kbyte dan diletakkan ke dalam SRAM dengan alamat dasar $0100. Dilanjutkan dengan melakukan pembacaan data letak tujuan pengkopian. Setelah letak tujuan pengkopian didapatkan data yang telah disimpan dalam SRAM dikirim menuju *flash drive* target. Demikian seterusnya hingga proses pengkopian berakhir.

4.4.2. Pengecekan Tombol Keypad

Pengecekan tombol ini dilakukan dengan cara melakukan proses scanning logika 0 pada urutan kolom tombol yang didapatkan dari pemberian logika 0 pada urutan baris.



Gambar 4.7. Diagram alir pengecekan tombol

4.4.3. Inisialisasi UART - SPI

Pada alat ini digunakan dua jalur komunikasi serial yakni jalur UART dan jalur SPI. Jalur UART digunakan untuk melakukan transaksi data antara VNC1L–1A dengan ATmega64, sedang SPI digunakan untuk melakukan transaksi data dari ATmega64 menuju LCD Nokia 3310.

4.4.4.1. Inisialisasi UART

Proses transaksi data antara ATmega64 dengan VNC1L–1A dilakukan pada kecepatan 2Mbps. Dengan demikian untuk mendapatkan kecepatan sebesar itu dari kristal 16 MHz, maka digunakan UART asinkron mode kecepatan ganda {U2X (UCSRA) = 1}, dimana nilai UBRR dapat dihitung sebagai berikut:

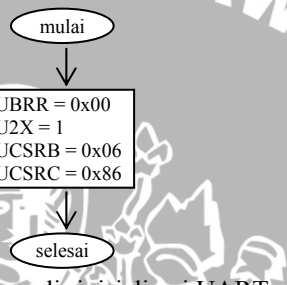
$$UBRR = \frac{16M}{8*2M} - 1 = 0$$

Proses selanjutnya meng-*enable* TXEN dan RXEN pada register UCSRB. Proses pengiriman dan penerimaan data dilakukan dengan format 8 bit data

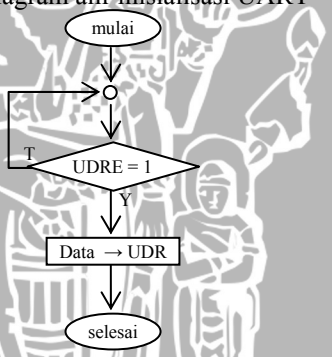
dengan 1 stop bit, dengan demikian bit UCSZ0 dan UCSZ1 pada register UCSRC diberi logika 1 dengan USBS bernilai logika 0.

Untuk menghindari penumpukan data pada saat pengiriman maka setiap kali akan menempatkan data pada register UDR harus melakukan pengecekan bit UDRE pada register UCSRA. Jika UDRE bernilai 1, maka register UDR telah kosong dan siap diisi dengan data yang baru.

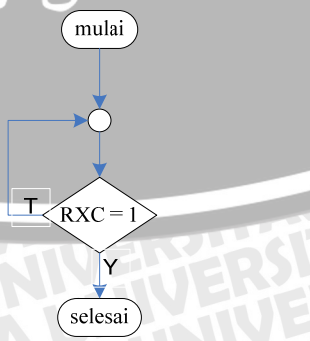
Untuk proses penerimaan data dilakukan dengan mengecek bit RXC pada register UCSRA, bila RXC benilai 1 maka data pada buffer penerima siap dibaca.



Gambar 4.8. Diagram alir inisialisasi UART



Gambar 4.9. Diagram alir kirim UART

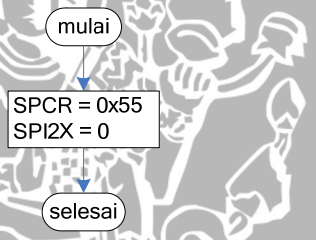


Gambar 4.10. Diagram alir terima UART

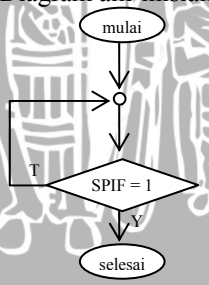
4.4.4.2. Inisialisasi SPI

Untuk proses transaksi data dari ATmega64 menuju ke LCD Nokia 3310, disesuaikan dengan kemampuan dan spesifikasi LCD Nokia 3310. LCD Nokia 3310 ini mempunyai spesifikasi SPI sebagai berikut kemampuan *clock* yang mencapai 4 MHz, dan tranfer dimulai dari MSB lebih dulu. Dengan demikian karena mikrokontroller ini manggunakan *clock* ekternal berupa kristal 16 Mhz, maka untuk mendapatkan frekuensi sebesar 0,5 MHz, SPI2X (SPSR), SPR1, dan SPR0 diberi logika 1, 1, dan 0. dan DORD diberi logika 0 supaya data pertama yang dikirim merupakan MSB.

Untuk pengecekan akhir pertukaran data dilakukan dengan mengecek bendera SPI (SPIF) pada register SPSR. Saat SPIF berlogika 1 maka proses pentranferan data telah berakhir, sebaliknya saat SPIF berlogika 0 maka masih terjadi proses pentranferan data.



Gambar 4.11. Diagram alir inisialisasi SPI

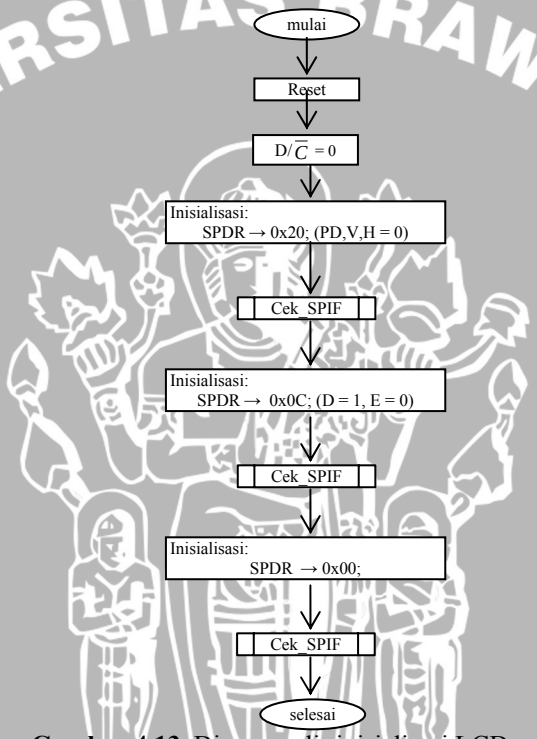


Gambar 4.12. Diagram alir Cek_SPIF

4.4.4. Tampilan dalam LCD

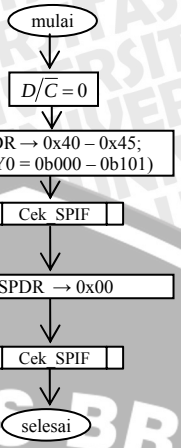
Pemberian tampilan dalam LCD Nokia 3310 ini dilakukan dengan mengirimkan perintah melalui jalur komunikasi SPI. Pengiriman perintah diawali dengan me-*reset* LCD kemudian memberikan logika nol pada bit D/$\overline{C}$.

Setelah di-*reset* LCD berada dalam keadaan *power down mode* (PD=1) dengan kondisi pengalamatan horisontal (V=0). Karena alasan tersebut maka perlu dilakukan inisialisasi agar chip dalam keadaan aktif yakni, dengan memberikan logika nol pada bit PD. Untuk merubah kondisi "*display blank*" setelah LCD direset menjadi dalam kondisi "*normal mode*", maka perlu merubah bit D dari logika 0 ke logika 1. Untuk merubah bit D ini dapat dilakukan dengan "*basic instruction set*" yakni dengan memberikan logika 0 pada bit H. Selanjutnya LCD telah siap dipakai untuk menampilkan data yang ingin di tampilkan.

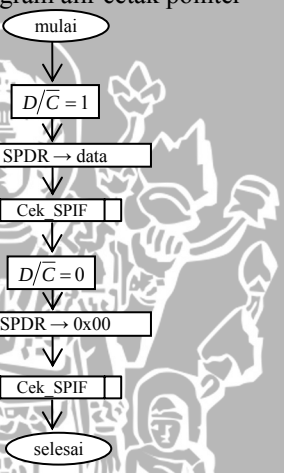


Gambar 4.13. Diagram alir inisialisasi LCD

Untuk mencetak karakter yang ingin ditampilkan di LCD maka bit $D/\overline{C}$ diberi logika 1, kemudian baru mengirimkan data yang ingin ditampilkan. Untuk mencetak pada pointer baris tertentu maka sebelum bit $D/\overline{C}$ dikenai logika 1, Y2:Y0 diberi nilai mulai dari 0b000 hingga 0b101 (baris 0 hingga baris 5). Setelah data dikirimkan kemudian $D/\overline{C}$ dikenai kembali logika nol (kembali ke mode perintah) dengan tujuan agar tidak ada data kosong yang dikirimkan setelah data terakhir.



Gambar 4.14. Diagram alir cetak pointer



Gambar 4.15. Diagram alir kirim data

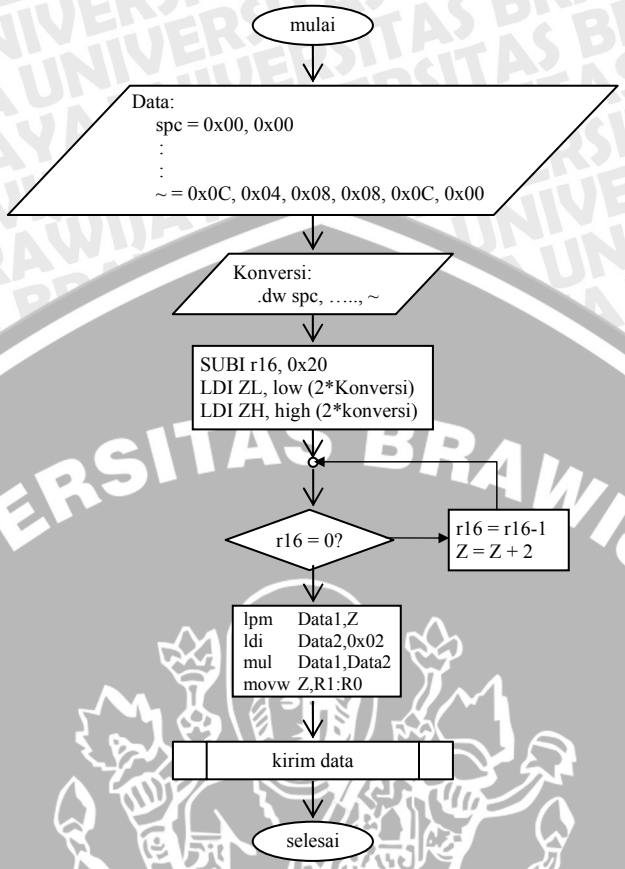
4.4.5. Konversi karakter ASCII ke karakter tampilan LCD

Pengkonversian ini dilakukan agar data yang diterima dari vinculum VNC1L–1A yakni, berupa kode ASCII dalam heksa desimal, dapat ditampilkan sebagai mana semestinya sehingga dapat dengan mudah dibaca oleh pengguna melalui media LCD. Nilai kode ASCII (*American Standart Code for Information Interchange*), dalam desimal (sebelah kiri) dan heksa (sebelah kanan) dijabarkan dalam Tabel 4.2.

Tabel. 4.2 Kode ASCII

32 20 (spc)	33 21 !	34 22 "	35 23 #	36 24 $	37 25 %	38 26 &	39 27 '
40 28 (	41 29)	42 2A *	43 2B +	44 2C ,	45 2D -	46 2E .	47 2F /
48 30 0	49 31 1	50 32 2	51 33 3	52 34 4	53 35 5	54 36 6	55 37 7
56 38 8	57 39 9	58 3A :	59 3B ;	60 3C <	61 3D =	62 3E >	63 3F ?
64 40 @	65 41 A	66 42 B	67 43 C	68 44 D	69 45 E	70 46 F	71 47 G
72 48 H	73 49 I	74 4A J	75 4B K	76 4C L	77 4D M	78 4E N	79 4F O
80 50 P	81 51 Q	82 52 R	83 53 S	84 54 T	85 55 U	86 56 V	87 57 W
88 58 X	89 59 Y	90 5A Z	91 5B [	92 5C \	93 5D]	94 5E ^	95 5F _
96 60 `	97 61 a	98 62 b	99 63 c	100 64 d	101 65 e	102 66 f	103 67 g
104 68 h	105 69 i	106 6A j	107 6B k	108 6C l	109 6D m	110 6E n	111 6F o
112 70 p	113 71 q	114 72 r	115 73 s	116 74 t	117 75 u	118 76 v	119 77 w
120 78 x	121 79 y	122 7A z	123 7B {	124 7C |	125 7D }	126 7E ~	127 7F □

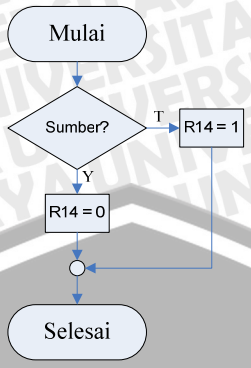
Pengkonversian ini pertama-tama dilakukan dengan cara menyusun data tampilan karakter dalam suatu tabel yang diurutkan berdasarkan urutan dalam kode ASCII seperti dalam Tabel 4.2. Kemudian melakukan pengurangan beberapa karakter yang tidak digunakan dalam menampilkan nama suatu file seperti: \, /, :, *, ?, “, <, >, dan | (heksa 0x5C, 0x2F, 0x 3A, 0x2A, 0x3F, 0x22, 0x3C, 0x3E, 0x7C) dengan tujuan menghemat memori flash. Selanjutnya untuk memanggil karakter yang ingin ditampilkan tinggal memanggil letak data karakter tersebut dalam tabel yakni dengan mengurangi bilangan heksa yang di dapat dari VNC1L–1A dengan bilangan heksa 0x20. Selanjutnya pointer akan bertambah secara otomatis dalam arah horisontal.



Gambar 4.16. Diagram alir konversi data

4.4.6. Pemilihan alamat penyimpanan untuk sumber dan target serta penentuan jenis perintah untuk sumber dan target

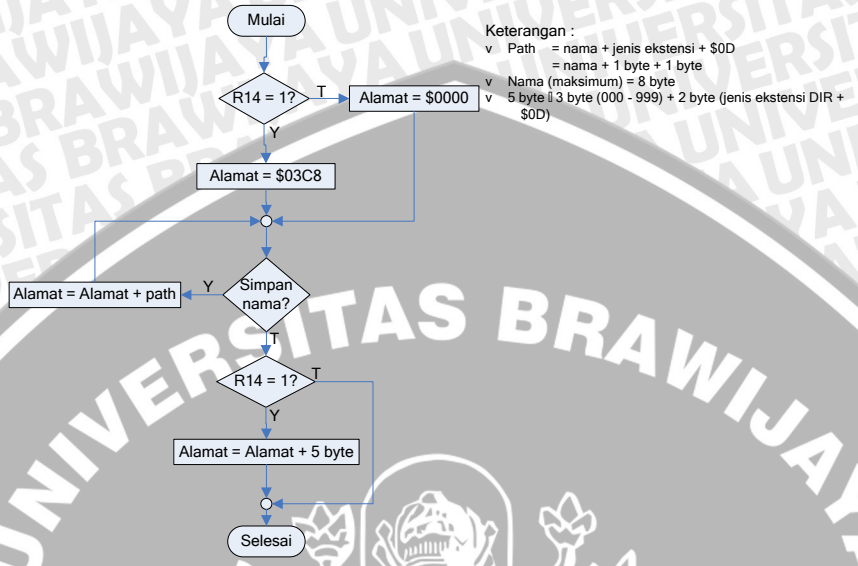
Penentuan jenis *flash drive* yang tertancap tergolong sumber atau target dilakukan dengan melihat register R14 bernilai logika 0 atau 1. R14 bernilai logika 0 bila yang tertancap merupakan *flash drive* sumber dan bernilai logika 1 bila yang tertancap merupakan *flash drive* target. Penentuan nilai logika R14 ini dilakukan saat penentuan sebuah port untuk dijadikan port sumber -, dalam hal ini port tersebut adalah port 1 atau port 2,- sedang port yang lain merupakan port target.



Gambar 4.17. Diagram alir pemilihan port sumber

Nilai R14 digunakan untuk menentukan letak penyimpanan *path file* dan level file dalam EEPROM, penentuan jenis perintah yang akan dikirimkan ke *flash drive* sumber atau target serta untuk pemilihan peletakan tampilan dalam LCD. *Path file* sumber diletakkan pada alamat dasar $0000 dan target pada alamat dasar $03B8. *Path file* ini berisikan nama file beserta jenis ekstensi yang mengikutinya, yakni apakah jenis file direktori atau jenis yang lain. Untuk jenis file direktori, perintah yang dikirimkan ke *flash drive* sumber (saat sebuah direktori dipilih) adalah "CD" kemudian spasi diikuti nama file direktori tersebut tanpa disertai ekstensi dari file tersebut, sedang untuk *flash drive* target, perintah yang dikirimkan ditambahi dengan sebuah perintah lagi, yakni "MKD" kemudian spasi diikuti nama file direktori tersebut tanpa disertai ekstensi dari file tersebut. Untuk jenis file yang lain, hanya dapat dilakukan setelah tombol pengkopian di tekan. Untuk jenis file ini, perintah yang dikirimkan ke *flash drive* sumber adalah "OPR" kemudian spasi diikuti nama file beserta ekstensinya atau "RDF" kemudian spasi diikuti dengan ukuran pembacaan data file, sedang untuk *flash drive* target perintah yang dikirimkan adalah "OPW" kemudian spasi diikuti nama file beserta ekstensinya atau "WRF" kemudian spasi diikuti dengan ukuran penulisan data file dilanjutkan dengan pengiriman data file yang telah tersimpan pada SRAM. Khusus untuk *flash drive* target pada pengaksesan *path* ditambahi dengan proses pengaksesan path sumber setelah trombol kopi ditekan. Jumlah level direktori untuk *flash drive* sumber terletak pada EEPROM dengan alamat $07F8, sedang untuk *flash drive* target

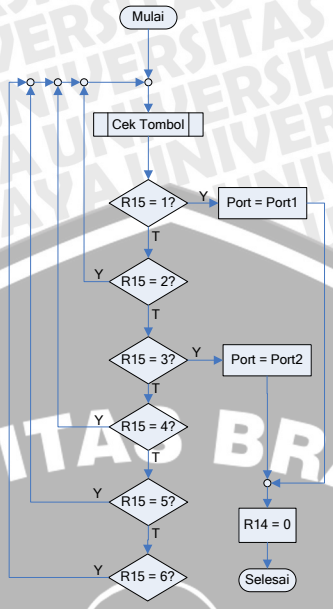
mempunyai alamat $07F9. Untuk jumlah level direktori sumber yang diakses sebelum tombol "*COPY*" ditekan dicatat pada EEPROM dengan alamat $07FA



Gambar 4.18. Diagram alir pemilihan penyimpanan *path* file

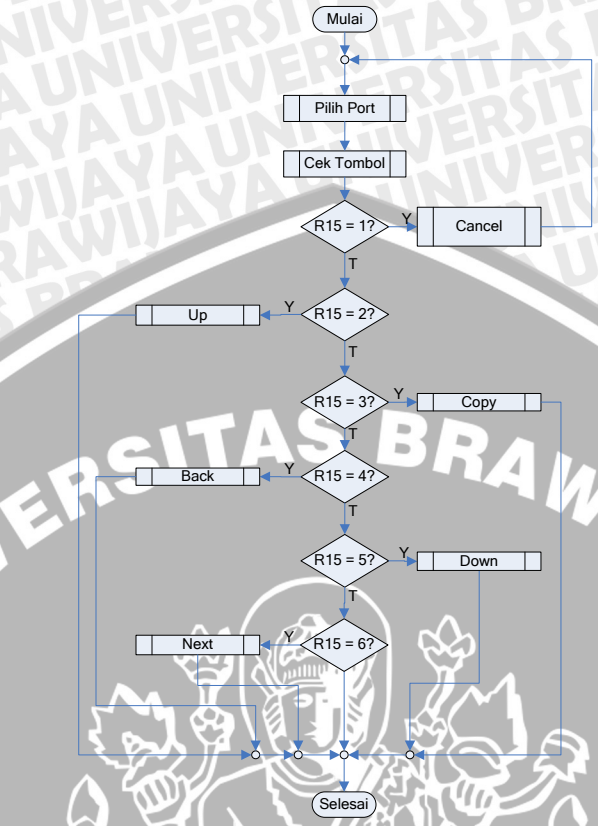
4.4.7. Pemilihan port sumber dan target serta pemilihan operasi *UP*, *DOWN*, *NEXT*, *BACK*, *COPY* dan *CANCEL*

Pada alat ini pemilihan port dan pemilihan operasi dilakukan dengan mengecek hasil kembalian dari penekanan tombol, yakni apakah r15 bernilai 1, 2, 3, 4, 5 atau 6. Untuk pemilihan port, r15 memiliki tiga nilai, yakni 1, 2 dan 3. Bila r15 bernilai 1, maka port 1 merupakan port untuk sumber sedang port 2 merupakan port untuk target. Bila r15 bernilai 3 maka berlaku hal sebaliknya port 2 bertindak sebagai sumber dan port 1 bertindak sebagai target. Bila r15 bernilai 2 maka proses dikembalikan ke pengecekan port (keluar dari proses pemilihan port). Untuk nilai yang lainnya proses kembali ke pengecekan penekanan tombol.



Gambar 4.19. Diagram alir pemilihan *port* sumber

Untuk pemilihan operasi, r15 memiliki enam nilai, yakni 1, 2, 3, 4, 5 dan 6. Bila r15 bernilai 1 maka proses yang dilakukan adalah proses "*CANCEL*" yakni pembatalan proses operasi pada *flash drive* tersebut. Bila r15 bernilai 2 maka proses yang dilakukan adalah proses "*UP*" yakni menggeser pemilihan file pada file yang terletak di atasnya berdasarkan urutan letak file dalam sebuah direktori. Bila r15 bernilai 3 maka proses yang dilakukan adalah proses "*COPY*" yakni melakukan penguncian letak file untuk melakukan proses pengkopian data.Bila r15 bernilai 4 maka proses yang dilakukan adalah proses "*BACK*" yakni menaikkan satu level direktori (dari sub-direktori menuju direktori *parent*) pada *flash drive* yang sedang di akses. Bila r15 bernilai 5 maka proses yang dilakukan adalah proses "*DOWN*" yakni menggeser pemilihan file pada file yang terletak di bawahnya berdasarkan urutan letak file dalam sebuah direktori. Bila r15 bernilai 6 maka proses yang dilakukan adalah proses "*NEXT*" yakni menurunkan satu level direktori (dari direktori *parent* menuju sub-direktori) pada *flash drive* yang sedang di akses.



Gambar 4.20. Diagram alir pemilihan operasi perintah

Pelaksanaan proses "*NEXT*" dilakukan dengan cara mengirimkan perintah CD diikuti nama file direktori yang dipilih kemudian meminta listing file dalam direktori file yang dipilih sebelumnya yakni, dengan mengirimkan perintah DIR diikuti nama file direktori tersebut. Sesaat listing diterima langsung dimasukkan ke dalam SRAM dengan alamat dasar $100. Pelaksanaan proses "*BACK*" hampir sama dengan proses "*NEXT*" yang berbeda adalah perintah awal yang dikirimkan yakni "CD..". Pelaksanaan proses "*UP*" dan "*DOWN*" dilakukan dengan menampilkan listing yang ada dalam SRAM ke dalam LCD secara satu persatu berdasarkan urutan letak file dalam file direktori mulai dari urutan yang teratas. Pelaksanaan proses "COPY" pada flash drive target terdapat sedikit perbedaan yakni buksn hanya mengunci nama file yang terakhir saja tetapi juga membuat file direktori baru. Perintah yang dikirimkan untuk pembuatan file direktori adalah MKD diikuti dengan nama file direktori. Penamaan file direktori ini dilakukan dengan cara mengirimkan kode ASCII untuk angka

desimal, sehingga bila terdapat nama file yang sama maka secara otomatis angka desimal tersebut ditambahkan hingga memiliki nama yang berbeda dengan nama file sub-direktori yang sudah ada pada direktori parent yang sedang diakses.

4.4.8. Proses mengetahui ukuran file yang dikopi dan ketersediaaan ruang dalam *flash drive* target

Proses mengetahui ukuran file yang dikopi dan ketersediaan ruang dalam *flash drive* target dimaksudkan untuk menghindari proses pengkopian yang berjalan tidak utuh. Untuk mengetahui ukuran file yang akan dikopi dilakukan dengan cara mengirimkan perintah DIR diikuti nama file tersebut. Sedang untuk ketersediaan ruang dalam *flash drive* target diketahui dengan mengirimkan perintah FS. Setelah mendapatkan ukuran file dan ketersediaan ruang, langkah selanjutnya membandingkan kedua hasil tersebut. Bila ukuran file lebih besar dari pada ketersediaan ruang maka proses pengkopian dihentikan, sedang bila ukuran file lebih kecil atau sama dengan ketersediaan ruang maka proses pengkopian dilaksanakan.

4.4.9. Proses pengkopian

Proses pengkopian pada alat ini dilakukan dengan cara mengambil data dari file dengan ukuran maksimum 3072 byte atau kurang lebih 3 kbyte. Bila data yang akan dikopi ternyata lebih besar dari pada 3072 byte maka data di ambil per 3072 byte dari sejumlah data yang akan dikopi. Data yang diambil dari *flash drive* sumber diletakkan ke dalam SRAM dengan alamat dasar $100. Setelah data didapat maka port USB dipindahkan ke port *flash drive* target untuk menuliskan data tersebut ke *flash drive* target. Untuk proses penggeseran letak pointer pembacaan awal dilakukan dengan mengirimkan perintah SEK diikuti ukuran pembacaan file sebelumnya.

BAB V
PENGUJIAN dan ANALISIS SISTEM

Tujuan pengujian sistem adalah untuk menentukan apakah sistem hasil rancangan ini berfungsi dengan baik dan sesuai dengan perancangan yang diinginkan. Selain itu untuk menentukan kondisi rangkaian pada masing-masing blok yang ada. Apabila rangkaian telah selesai dirancang dan dibuat, maka rangkaian ini harus diuji terlebih dahulu untuk memastikan alat ini dapat dioperasikan sesuai dengan tujuan perancangan yang diinginkan. Pengujian dan pengamatan dilakukan terhadap sistem pengkabelan, hubungan tiap komponen, tiap blok rangkaian dan pengujian sistem keseluruhan.

Pada bab ini akan dilakukan juga pembahasan dari setiap pengujian dan pengamatan yang dilakukan. Pengujian ini dilakukan pada tiap-tiap blok dalam sistem dengan mengikuti prosedur yang telah dijelaskan dalam Bab III, yaitu:

1.) Pengujian Blok LCD Nokia 3310
2.) Pengujian Blok Vinculum VDIP2
3.) Pengujian Keseluruhan Sistem

Alat-alat bantu yang digunakan dalam pengujian ini menggunakan beberapa instrumen diantaranya adalah sebagai berikut:

- Catu daya DC + 5V
- Multimeter digital UNI-T UT70A
- PC (*Personal Computer*)

5.1. Pengujian Blok LCD Nokia 3310

5.1.1. Tujuan

Tujuan pengujian ini adalah untuk mengetahui apakah LCD dapat menampilkan karakter yang ingin ditampilkan dengan baik atau tidak.

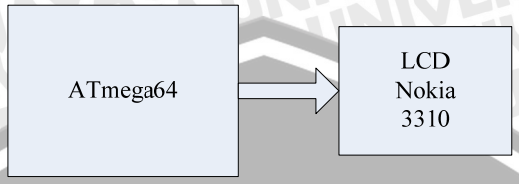
5.1.2. Prosedur Pengujian

Langkah – langkah yang diambil untuk menguji blok LCD Nokia 3310 adalah sebagai berikut:

1.) Menyusun rangkaian seperti terlihat dalam Gambar 5.1

2.) Menyalakan catu daya

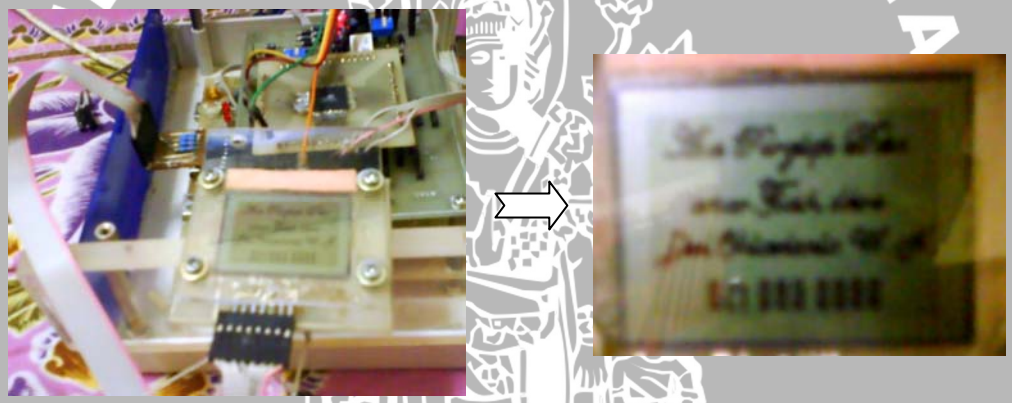
3.) Melihat tampilan data yang telah dimasukkan ke dalam ATmega64 sebelumnya pada layar LCD



Gambar 5.1. Blok diagram pengujian blok LCD Nokia 3310

5.1.3. Hasil Pengujian dan Analisis Data

Dari hasil pengujian yang dapat dilihat pada Gambar 5.2 (b) dapat diketahui bahwa blok LCD Nokia 3310 bekerja sesuai dengan apa yang diharapkan.



(a) (b)

Gambar 5.2. Gambar pengujian (a) dan hasil uji (b) coba blok LCD Nokia 3310

5.2. Pengujian Blok Vinculum VDIP2

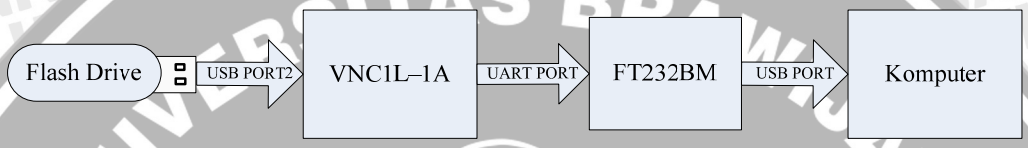
5.2.1. Tujuan

Tujuan pengujian ini adalah untuk mengetahui apakah blok Vinculum DIP2 tersebut bekerja dengan baik apa tidak, yakni mampu untuk digunakan sebagai antarmuka dengan *flash drive* dengan baik apa tidak serta untuk mengetahui apakah jalur komunikasi UART dapat berfungsi dengan baik atau tidak.

5.2.2. Prosedur Pengujian

Langkah – langkah yang diambil untuk menguji blok Vinculum VDIP2 adalah sebagai berikut:

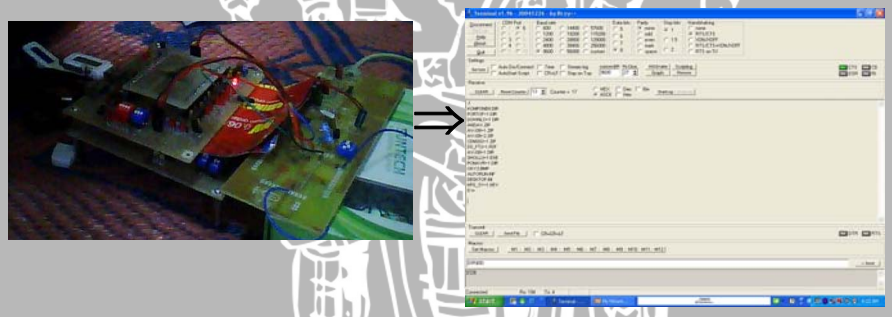
1.) Menyusun rangkaian seperti Gambar 5.3
2.) Mengaktifkan program "Terminal“ di komputer.
3.) Mengirimkan perintah pada kotak perintah di program "Terminal“
4.) Melihat hasil balik dari Vinculum VNC1L–1A pada kotak tampilan di program "Terminal“



Gambar 5.3. Blok diagram pengujian modul VNC1L–1A

5.2.3. Hasil Pengujian dan Analisis Data

Dari hasil pengujian yang dapat dilihat pada Tabel 5.1 dapat diketahui bahwa modul sistem VNC1L–1A bekerja sesuai dengan apa yang diharapkan.



(a) (b)

Gambar 5.4. Gambar pengujian modul VNC1L–1A (a) dengan hasil pengujian (b)

Tabel. 5.1 Hasil pengujian modul VNC1L–1A

PERINTAH	HASIL	KETERANGAN
$0D	No Upgrade	Pertama kali *flash drive* tertancap
MKD OKY↵	D:\>	Buat file direktori dengan nama “OKY”
DIR↵	WINRAR WINDOW~1.EXE WINRAR~1.ZIP WINZIP~1.ZIP OKY DIR D:\>	Data semua file yang ada dalam root direktori
CD.. ↵	Bad Command	Perintah salah
CD .. ↵	Command Failed	Karena berada pada *root* direktori
DIR OKY↵	OKY <0><0><0><0> D:\>	

CD OKY↵	D:\>	Masuk ke dalam direktori “OKY”
OPW NJAJAL.TXT↵	D:\>	Buat file “NJAJAL.TXT”
WRF 4↵ OKI	D:\>	Tulis 3 byte (“OKI”)
CLF NJAJAL.TXT↵	D:\>	Tutup file “NJAJAL.TXT”
OPR NJAJAL.TXT↵	D:\>	Buka file “NJAJAL.TXT”
RD NJAJAL.TXT↵	OKI	Baca keseluruhan isi file
CLF NJAJAL.TXT↵	D:\>	Tutup file “NJAJAL.TXT”
OPW NJAJAL.TXT↵	D:\>	Buat file “NJAJAL.TXT”
WRF 10↵ NJAJAL1234	D:\>	Tulis 3 byte (“NJAJAL1234”)
CLF NJAJAL.TXT↵	D:\>	Tutup file “NJAJAL.TXT”
OPR NJAJAL.TXT↵	D:\>	Buka file “NJAJAL.TXT”
RD NJAJAL.TXT↵	OKI NJAJAL1234	Baca keseluruhan isi file
CLF NJAJAL.TXT↵	D:\>	Tutup file “NJAJAL.TXT”
DIR↵	NJAJAL.TXT D:\>	Data semua file yang ada dalam file direktori “OKY”
CD .. ↵	D:\>	Naikkan direktori satu level
DIR↵	WINRAR WINDOW~1.EXE WINRAR~1.ZIP WINZIP~1.ZIP OKY DIR D:\>	Data semua file yang ada dalam root direktori

5.3. Pengujian Blok Keseluruhan Sistem

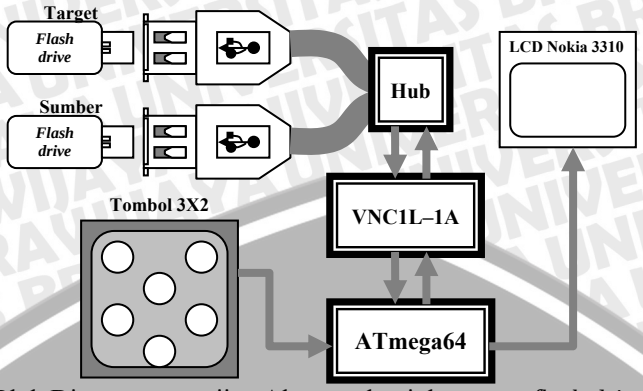
5.3.1. Tujuan

Pengujian ini dilakukan untuk mengetahui unjuk kerja alat telah bekerja dengan baik atau tidak.

5.3.2. Prosedur Pengujian

Langkah – langkah yang diambil untuk menguji blok keseluruhan sistem adalah sebagai berikut:

1.) Menyusun rangkaian seperti Gambar 5.6
2.) Menyalakan catu daya
3.) Melihat tampilan pada LCD
4.) Menekan tombol
5.) Melihat tampilan pada LCD
6.) Melakukan proses pembacaan *flash drive* sumber dan target
7.) Melakukan proses pengkopian



Gambar 5.5. Blok Diagram pengujian Alat pengkopi data antar *flash drive* menggunakan Atmega64 dengan antar-muka VNC1L–1A

5.3.3. Hasil Pengujian dan Analisis Data

Dari hasil pengujian tampak bahwa blok Atmega64 telah berhasil berkomunikasi dengan baik dengan blok LCD Nokia3310 dan blok Vinculum VDIP2.

Gambar 5.6. Pengujian Alat pengkopi data antar *flash drive* menggunakan Atmega64 dengan antar-muka VNC1L–1A

BAB VI
PENUTUP

6.1. Kesimpulan

Dari hasil pengujian alat yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

1.) Mikrokontroller ATmega64 dan VNC1L–1A dapat digunakan sebagai komponen utama alat pengkopi data antar *flash drive*.
2.) Alat pengkopi data antar *flash drive* ini mampu membantu proses pentransaksian data antar *flash drive*, yang dalam hal ini dipusatkan pada masalah pengkopian data secara efektif dan efisien.

6.2. Saran

Untuk pengembangan alat selanjutnya ada beberapa hal yang dapat dipertimbangkan, antara lain:

1.) Penggunaan komponen SMD, untuk penyederhanaan desain ruang
2.) Penanganan file ganda, dikarenakan alat yang saya buat ini membuat direktori baru setiap satu proses pengkopian baik folder maupun file.
3.) Penanganan waktu proses pengkopian yang relatif lama dikarenakan proses perpindahan port menggunakan relay.

Daftar Pustaka

Amontec. 2004 , *Nokia 3310 LCD 84x48* pixels http://www.amontec.com/lcd_nokia_3310.shtml (diakses : Rabu, 18 April 2007, pukul: 9:45:06)

Atmel Corporation, 2003, *ATmega64(L) Preliminary* http://www.atmel.com/dyn/resources/prod_documents/doc2490.pdf (diakses: Rabu, 31 Januari 2007, pukul: 12:11:24)

Future Technology Devices International Ltd, 2006, *Vinculum VNC1L Embedded USB Host Controller I.C.* Datasheet Version 0.95 http://www.vinculum.com/documents/datasheets/DS_VNC1L-1A.pdf (diakses : Minggu, 21 January 2007, pukul: 3:43:09)

Future Technology Devices International Ltd. (FTDI), 2007, *Future Technology Devices International Ltd Vinculum Firmware User Manual* Version: 2.1 *l,* Glasgow, UK

2.2.5 Pemrograman VNC1L

Agar dapat digunakan IC VNC1L-1A ini harus terlebih dahulu diisi dengan salah satu program yang telah disediakan oleh FTDI dalam situsnya http://www.vinculum.com/downloads.html, antara lain:

- VDAP *Disk And Peripheral Firmware*
- VMSC *Music Firmware*
- VDPS *Disk or Peripheral Firmware*
- VCDC *Communication Device Class Firmware*
- VDIF *Disk Interface Firmware*
- VDFC *Disk File Copy Firmware*
- VF2F *Firmware*

Firmware yang disediakan tersebut di-*upload*-kan ke dalam IC VNC1L–1A dengan perantara IC FT232BM/FT232BL menggunakan software

"VPROG" yang telah disediakan dalam situs vinculum. Sebelum meng–*upload* program, IC VNC1L–1A di-*set* terlebih dahulu ke mode komunikasi UART, serta menghubungkan pin10 (PROG#) ke ground. *Default* UART VNC1L–1A adalah baudrate sebesar 9600, 8 bit data, 1 bit *start*, 1 bit *stop*, dan tanpa paritas dengan sinyal handshaking RTS/CTS aktif.

Untuk merubah nilai default tersebut dapat dilakukan dengan mengaplikasikan software "VncFwMod" yang juga telah disediakan oleh vinculum dalam situs-nya atau melalui mode perintah (*command mode*). http://www.compsys1.com/support/vnc1/VinculumFirmwareUserManual.pdf (diakses : : Senin, 24 September 2007, pukul : 9:57:23)

Heybruck, William F., Dr., 2005, *An Introduction to FAT 16/FAT 32 File Systems,* Hitachi Global Storage Technologies, Charlotte, NC http://hitachigst.com/tech/techlib.nsf/techdocs/BB4945CEAAE4DAD986256D890016E8F4/$file/FAT_White_Paper_FINAL.pdf (diakses : Kamis, 19 Juli 2007, pukul : 24:31:12)

Leong, Chui Wei, 2005, *Understanding Universal Serial Bus (USB),* USBDeveloper www.usbdeveloper.com/UnderstandUSB/understandusb.htm (diakses : Selasa, 27 Juni 2006, pukul : 15:11:40)

Microsoft Corporation, 2000, *Microsoft Extensible Firmware Initiative FAT32 File System Specification FAT: General Overview of On-Disk Format*

http://staff.washington.edu/dittrich/misc/fatgen103.pdf (diakses : Sabtu, 27 Januari 2007, pukul: 20:30:49)

NTFS, 2006, *File Allocation System* www.ntfs.com/fat-allocation.htm (diakses : Sabtu, 27 Januari 2007, pukul: 22:30:54)

Peacock, Craig, 2002,USB *in a Nutshell: Making Sense of the USB Standard, Beyond Logic* http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf (diakses : Selasa, 27 Juni 2006, pukul : 15:07:26)

Philips Semiconductors, 1999, *DATA SHEET: PCD8544 48 ´ 84 pixels matrix LCD controller/driver*, EINDHOVEN, The Netherlands, http://www.amontec.com/lcd_controller_pcd8544.pdf (diakses : Senin, 16 April 2007, pukul : 8:55:14)

Stoffregen Paul, 2005,Understanding *FAT32 Filesystems*, PJRC http://www.pjrc.com/tech/8051/ide/fat32.html (diakses : Kamis, 19 Juli 2007, pukul : 24:35:06)

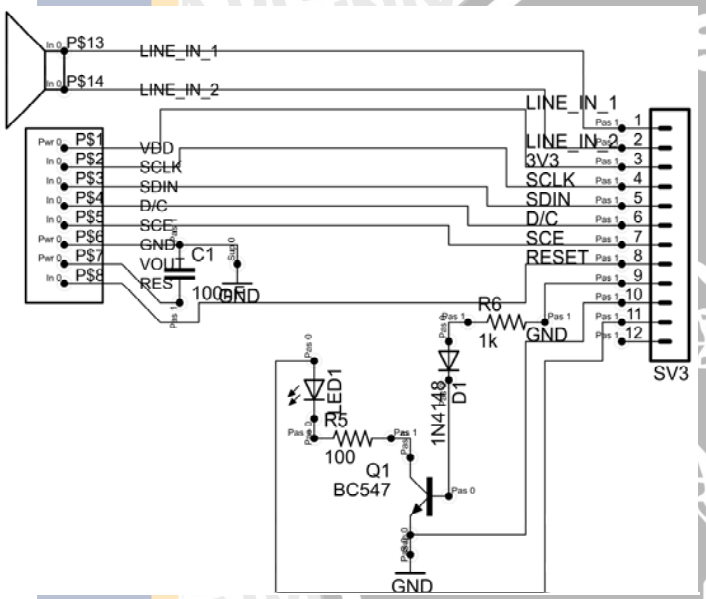
Sutadi, Dwi. 2004. *I/O BUS & Motherboard*. ANDI. Yogyakarta

Transcend Information Inc., 2006, *TS512MJF110: 512MB USB2.0 JetFlash™* http://www.transcendusa.com/Support/DLCenter/Datasheet/512MJF110.pdf (diakses : Kamis, 19 Juli 2007, pukul : 24:57:19)

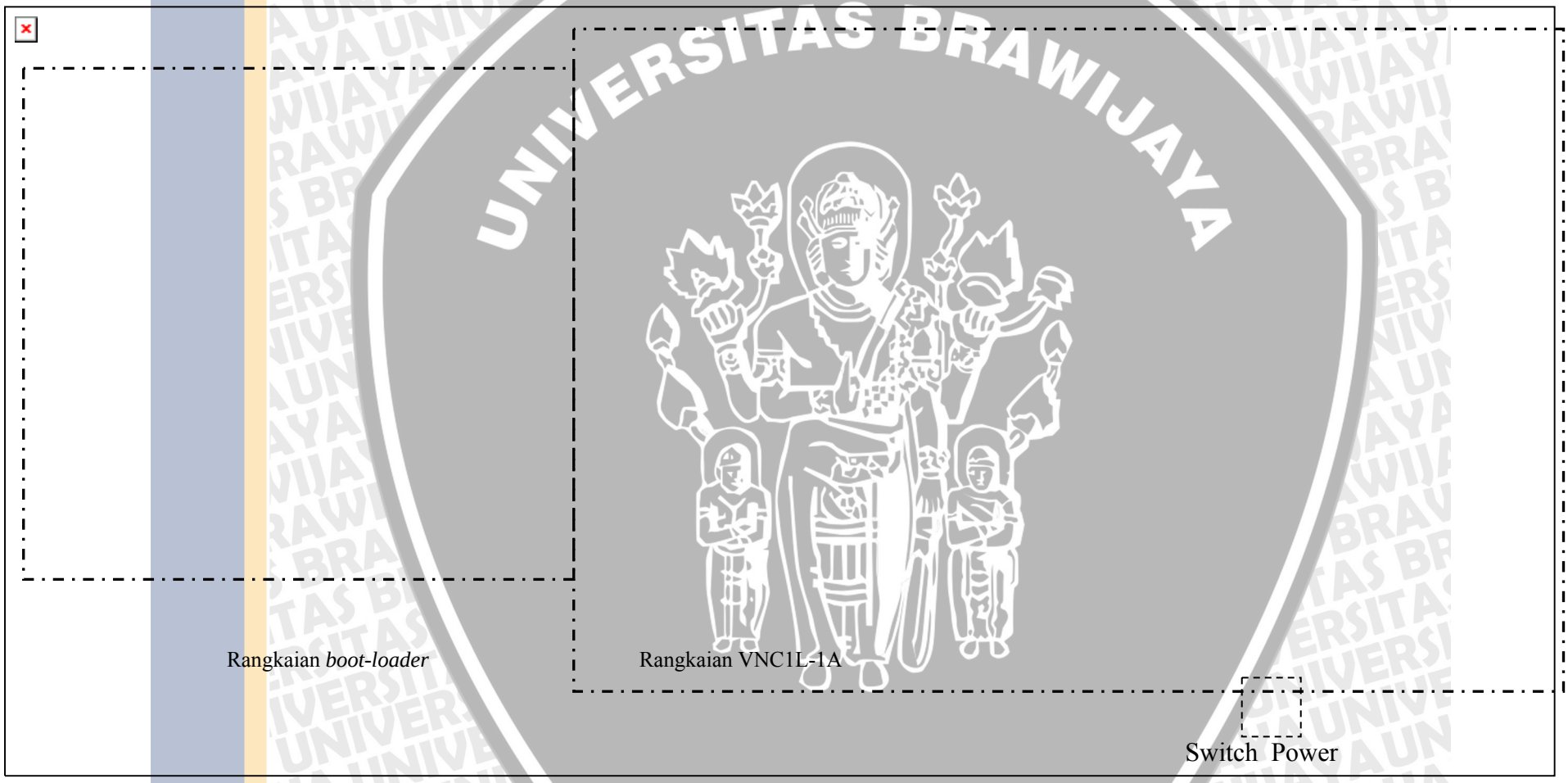
LAMPIRAN

Lampiran 1:

Blok LCD Nokia 3310



Lampiran 2:
Blok VNC1L–1A



Lampiran 3:
Blok ATmega64

Lampiran 2:

Listing program "Alat Pengkopi Data Antar *Flash Drive* Menggunakan ATmega 64 dengan Antar-muka VNC1L"

```
//*****************************************************************//
//                          Program Utama                          //
//*****************************************************************//

.include "m64def.inc"
.include "deklarasi.asm"
.include "LCD Nokia 3310.asm"
.include "penundaan.asm"
.include "pilih_alamat.asm"
.include "VNC1L-1A.asm"
.include "Tombol.asm"
.include "memori.asm"
.include "proses.asm"

Utama:
   ldi      Data1,0b11111111
   out      ddrb,Data1
//brstmbl2,brstmbl1,backlight,DATA/Comman
d(1/0),Reset,MOSI,SCK,relay//
   sbi      portb,3
   ldi      Data1,0b00011001
   out      ddrd,Data1
//klmtmbl3,klmtmbl2,klmtmbl1,RTS(low),TX
D,RXD,CTS(low),relay//
   ldi      Data1,0b11100010
   out      portd,Data1
   sbi      ddre,7
//brstmbl2//
   ldi      Data1,high(ramend)
   out      SPH,Data1
   ldi      Data1,low(ramend)
   out      SPL,Data1

// Inisialisasi UART = 2 Mbps
   sbr      Data1,(1<<u2X1)
   sts      UCSR1A,Data1
   ldi      Data1,(3<<UCSZ10)
   sts      UCSR1C,Data1
   ldi      Data1,high(nilai_ubbr)
   sts      UBRR1H,Data1
   ldi      Data1,low(nilai_ubbr)
   sts      UBRR1L,Data1
   ldi      Data1,(1<<RXEN1)|(1<<TXEN1)
;enable rcvr and trmttr
   sts      UCSR1B,Data1

// Inisialisasi SPI = 0.25 Mbps
   ldi      Data1,(1<<SPI2X)
   out      SPSR,Data1
   ldi   Data1,(1<<SPE)|(1<<MSTR)|(1<<CPHA)|(1<<CPOL)|(3<<SPR0)
   out      SPCR,Data1
;mereset LCD Nokia 3310
   cbi      portb,3
   ldi      Data2,0x01
   ldi      Data1,0x10
   call     TundaNOP
   sbi      portb,3

//inisialisasi LCD Nokia 3310
   ldi      ZH,high(init_LCD2*2)
   ldi      ZL,low(init_LCD2*2)
   call     prth_LCD
   ldi      ZH,high(init_LCD3*2)
   ldi      ZL,low(init_LCD3*2)
   call     prth_LCD
   ldi      ZH,high(init_LCD*2)
   ldi      ZL,low(init_LCD*2)
   call     prth_LCD

//Menyalakan Back light
   sbi      portb,5

//inisialisasi VNC1L-1A
   rcall    enter
lompat:
//versi firmware vinculum
   rcall    enter
```

```
//cetak nama ke dalam LCD Nokia 3310
   ldi     ZH,high(nama*2)
   ldi     ZL,low(nama*2)
   call    data_LCD

lompat1:
//cek port USB1
   cbi     portb,0
   rcall   cek_port1

//cek port USB2
   rcall   cek_port

//menu pilih port sumber
   ldi     ZH,high(plh_pt*2)
   ldi     ZL,low(plh_pt*2)
   call    data_LCD

//mengecek pilihan sumber
lompat2:
   call    cek_tbl

   mov     Data1,Tombol

//port USB1 adalah sumber
   cpi     Data1,0x01
   brne    lompat3
   call    cek_port
   rjmp    lompat5
lompat3:

//tidak jadi mengkopi
   cpi     Data1,0x02
   brne    lompat4
   call    cabut
   rjmp    lompat
lompat4:

//port USB2 adalah sumber
   cpi     Data1,0x03
   brne    lompat2

lompat5:
   call    hapus
   call    tunda

//set kode untuk sumber (0) atau target (1)
   clr     SatauT

//menulis sumber pada baris pertama
   ldi     ZH,high(LCD_baris1*2)
   ldi     ZL,low(LCD_baris1*2)
   call    prth_LCD

//cetak karakter S
   ldi     Data1,0x53
   call    ubah

//cetak karakter u
   ldi     Data1,0x75
   call    ubah

//cetak karakter m
   ldi     Data1,0x6D
   call    ubah

//cetak karakter b
   ldi     Data1,0x62
   call    ubah

//cetak karakter e
   ldi     Data1,0x65
   call    ubah

//cetak karakter r
   ldi     Data1,0x72
   call    ubah

//mulai proses pencarian direktori sumber
   call    crDIR

//cek apakah jadi
   cpi     Data1,0xff
   brne    lompat6
   call    cabut
   rjmp    lompat

//cek apakah disk penuh
   cpi     Data1,0xfe
   brne    lompat6
   rjmp    akhir

lompat6:
//ganti port ke target port
   inc     SatauT
   call    cek_port

//menulis target pada baris ke 3
   ldi     ZH,high(LCD_baris3*2)
   ldi     ZL,low(LCD_baris3*2)
   call    prth_LCD

//cetak karakter T
   ldi     Data1,0x54
   call    ubah

//cetak karakter a
   ldi     Data1,0x61
```

```
    call    ubah

//cetak karakter r
    ldi     Data1,0x72
    call    ubah

//cetak karakter g
    ldi     Data1,0x67
    call    ubah

//cetak karakter e
    ldi     Data1,0x65
    call    ubah

//cetak karakter t
    ldi     Data1,0x74
    call    ubah
    call    crDIR

//cek apakah disk penuh
    cpi     Data1,0xfe
    brne    lompat7
    rjmp    akhir

lompat7:
//cek apakah jadi
    cpi     Data1,0xff
    brne    lompat8
    call    cabut
    rjmp    lompat


lompat8:
    call    hapus
//tanda memulai proses pengkopian
    ldi     ZH,high(gambar1*2)
    ldi     ZL,low(gambar1*2)
    call    data_LCD

//kembali ke port sumber
    dec     SatauT
    call    cek_port
//lihat jumlah direktori yang digunakan
    ldi     XH,high(urut5)
    ldi     XL,low(urut5)
    call    bacaEEPROM
    dec     Data1
    mov     cpy,Data1

    ldi     XH,high(jjksmbr)
    ldi     XL,low(jjksmbr)
    ldi     Data2,0x00
lompat9:
    call    Flash
    inc     Data2
    cp      Data2,cpy
    brne    lompat9
lompat10:
//cek apakah file merupakan file "dir"
    ld      Data1,X
    cpi     Data1,0x02
    brne    lompat11
    rjmp    lompat12
lompat11:
    call    Flash
    call    Rekam
//tambah level pengkopian
    inc     cpy
    call    cekekstensi
//rekam jumlah nama file dalam direktori
    ldi     XH,high(urut1)
    ldi     XL,low(urut1)
    add     Xl,cpy
    ldi     Data1,0x00
    adc     XH,Data1
    mov     Data1,jmlurut
    call    tulisEEPROM
//rekam posisi urutan file dalam direktori
    ldi     XH,high(urut2)
    ldi     XL,low(urut2)
    add     Xl,cpy
    ldi     Data1,0x00
    adc     XH,Data1
    mov     Data1,nourut
    call    tulisEEPROM
//kembali ke pointer penjejak file
    movw X,pointerH:pointerL
    rjmp    lompat10
lompat12:
//rekam pointer awal X
    movw pointerH:pointerL,X
//tambahkan level pengkopian
//akibat pengurangan cpy sebelumnya
    inc     cpy
    mov     Data1,cpy
//rekam level pengkopian
    ldi     XH,high(urut5)
    ldi     XL,low(urut5)
    call    tulisEEPROM
//kembalikan level pengkopian
    movw X,pointerH:pointerL
//cek apakah ukuran file sudah 0x00 00 00 00
    ldi     Data1,0x00
    cp      ukur4,Data1
    brne    lompat15
    cp      ukur3,Data1
    brne    lompat15
    cp      ukur2,Data1
```

```
    brne    lompat15
    cp      ukur1,Data1
    brne    lompat15
//lihat ukuran file
    ldi     ZH,high(DIR_file*2)
    ldi     ZL,low(DIR_file*2)
    call    kirim_UART
    adiw    X,0x01
lompat13:
    call    bacaEEPROM
    call    CTS
    sts     UDR1,Data1
    adiw    X,0x01
    cpi     Data1,0x0D
    brne    lompat13
//kembalikan level pengkopian
    movw X,pointerH:pointerL
//lewati hingga dapat spasi (0x20)
lompat14:
    call    trm_UART
    cpi     Data1,0x20
    brne    lompat14
//rekam ukuran file
0x(ukur1)(ukur2)(ukur3)(ukur4)
    call    trm_UART
    mov     ukur4,Data1
    call    trm_UART
    mov     ukur3,Data1
    call    trm_UART
    mov     ukur2,Data1
    call    trm_UART
    mov     ukur1,Data1
//terima hingga prompt
    call    prmpt
//reset jumlah pengurangan
    clr     Kurang1
    clr     Kurang2
lompat15:
    call    kurang
    call    Flash

    call    hapus
//tanda sedang melakukan proses pengkopian
    ldi     ZH,high(tunggu*2)
    ldi     ZL,low(tunggu*2)
    call    data_LCD
//kembali ke port target
    inc     SatauT
    call    cek_port
    ldi     Data1,0x00
    cp      Kurang1,Data1
    brne    lompat17
    cp      Kurang2,Data1
    brne    lompat17
//lihat ruang kosong dalam flash drive
    ldi     ZH,high(FS*2)
    ldi     ZL,low(FS*2)
    call    krm_UART
//tempatkan besar kapasitas kosong ke SRAM
    ldi     YH,0x0E
    ldi     YL,0x00
    ldi     Data2,0x06
//kapasitas 0x(0E00)(0E01)(0E02)(0E03)(0E04)(0E05)
lompat16:
    call    trm_UART
    st      Y+,Data1
    call    trm_UART
    dec     Data2
    cpi     Data2,0x00
    brne    lompat16
//terima hingga prompt
    call    prmpt

//arahkan ke nilai MSB-nya (0E00)
    sbiw    Y,0x07
//cek apakah (0E00) = 0
    ld      Data1,Y+
    cpi     Data1,0x00
    brne    lompat17
//cek apakah (0E01) = 0
    ld      Data1,Y+
    cpi     Data1,0x00
    brne    lompat17
//cek apakah (0E02) = 0
    ld      Data1,Y+
    cp      ukur1,Data1
    brlo    lompat17
    cpse    ukur1,Data1
    rjmp    akhir
//cek apakah (0E03) = 0
    ld      Data1,Y+
    cp      ukur2,Data1
    brlo    lompat17
    cpse    ukur2,Data1
    rjmp    akhir
//cek apakah (0E04) = 0
    ld      Data1,Y
    cp      ukur3,Data1
    brlo    lompat17
//(0E05) tidak dicek
//karena ukuran sektor = 512
    rjmp    akhir
lompat17:
//lihat jumlah direktori sumber yang dibuka
    ldi     XH,high(urut3)
    ldi     XL,low(urut3)
    call    bacaEEPROM
    mov     sumber,Data1
```

```
//lihat jumlah direktori target yang dibuka
   ldi     XH,high(urut4)
   ldi     XL,low(urut4)
   call    bacaEEPROM
   mov     target,Data1
//lihat jumlah direktori yang digunakan
   ldi     XH,high(urut5)
   ldi     XL,low(urut5)
   call    bacaEEPROM
   dec     Data1
   mov     cpy,Data1
//buka penjejak target
   ldi     XH,high(jjktgt)
   ldi     XL,low(jjktgt)
   ldi     Data2,0x00
lompat18:
   call    Flash
   inc     Data2
   cp      Data2,target
   brne    lompat18
//geser penjejak sumber hingga posisi kopi
   ldi     XH,high(jjksmbr)
   ldi     XL,low(jjksmbr)
   ldi     Data2,0x01
lompat19:
   call    bacaEEPROM
   adiw    X,0x01
   cpi     Data1,0x0D
   brne    lompat19
   inc     Data2
   cp      Data2,sumber
   brne    lompat19
//kurangi penjejak sumber dari posisi kopi
   mov     Data2,cpy
   sub     Data2,sumber
lompat20:
   call    Flash
   dec     Data2
   cpi     Data2,0x00
   brne    lompat20
//cek ukuran file = 0x00 00 00 00
   ldi     Data1,0x00
   cpse    ukur1,Data1
   rjmp    lompat8
   cpse    ukur2,Data1
   rjmp    lompat8
   cpse    ukur3,Data1
   rjmp    lompat8
   cpse    ukur4,Data1
   rjmp    lompat8
//cek apakah penjejak sumber dari kopi =
0x01?
   dec     sumber
   mov     Data1,cpy
   sub     Data1,sumber
//cek apakah akhir kopi
   cpi     Data1,0x01
   brne    lompat21
   rjmp    akhir

lompat21:
   call    hapus
//tanda menggeser file yang dikopi
   ldi     ZH,high(gambar2*2)
   ldi     ZL,low(gambar2*2)
   call    data_LCD
//kembali ke port sumber
   dec     SatauT
   call    cek_port
lompat22:
//lihat jumlah direktori yang digunakan
   ldi     XH,high(urut5)
   ldi     XL,low(urut5)
   call    bacaEEPROM
   dec     Data1
   mov     cpy,Data1
//rekam jumlah nama file dalam direktori
   ldi     XH,high(urut1)
   ldi     XL,low(urut1)
   add     Xl,cpy
   ldi     Data1,0x00
   adc     XH,Data1
   call    bacaEEPROM
   mov     jmlurut,Data1
//rekam posisi urutan file dalam direktori
   ldi     XH,high(urut2)
   ldi     XL,low(urut2)
   add     Xl,cpy
   ldi     Data1,0x00
   adc     XH,Data1
   call    bacaEEPROM
   inc     Data1
   mov     nourut,Data1
   cp      nourut,jmlurut
   brne    lompat23
   dec     cpy
//catat jumlah direktori yang digunakan
   ldi     XH,high(urut5)
   ldi     XL,low(urut5)
   mov     Data1,cpy
   inc     Data1
   call    tulisEEPROM
   rjmp    lompat22
lompat23:
//rekam posisi urutan file dalam direktori
   ldi     XH,high(urut2)
   ldi     XL,low(urut2)
   add     Xl,cpy
```

```
    ldi     Data1,0x00
    adc     XH,Data1
    inc     nourut
    mov     Data1,nourut
    call    tulisEEPROM
    ldi     XH,high(jjksmbr)
    ldi     XL,low(jjksmbr)
    ldi     Data2,0x00
lompat24:
    call    Flash
    inc     Data2
    cp      Data2,cpy
    brne    lompat24
    call    Rekam
//rekam posisi
    movw pointerH:pointerL,X
//rekam posisi urutan file dalam direktori
    ldi     XH,high(urut2)
    ldi     XL,low(urut2)
    add     Xl,cpy
    ldi     Data1,0x00
    adc     XH,Data1
    call    bacaEEPROM
    mov     nourut,Data1
    call    cekekstensi
    call    hapus
//tanda menggeser file yang dikopi
    ldi     ZH,high(gambar3*2)
    ldi     ZL,low(gambar3*2)
    call    data_LCD
//kembali ke port target
    inc     SatauT
    call    cek_port
    rjmp    lompat8

akhir:
    call    hapus
//tanda menggeser file yang dikopi
    ldi     ZH,high(selesai*2)
    ldi     ZL,low(selesai*2)
    call    data_LCD
//cek flash drive sumber sudah dicabut
    clr     SatauT
    call    cabut
//cek flash drive sumber sudah dicabut
    inc     SatauT
    call    cabut

    //kembali ke menu utama
    call    hapus
    //cetak nama ke dalam LCD Nokia 3310
    ldi     ZH,high(nama*2)
    ldi     ZL,low(nama*2)
    call    data_LCD
    jmp     lompat1
//*****************************************************************//
//                      Akhir Program Utama                        //
//*****************************************************************//
```

```
//*****************************************************************//
//                          Deklarasi.asm                          //
//*****************************************************************//
//=================================================================//
//                      Deklarasi register                         //
//=================================================================//
// register untuk Flash (Z) R31 dan R30 //
// register untuk SRAM (Y)  R29 dan R28  //
// register untuk EEPROM (X) R27 dan.R26 //
// register posisi alamat   //
.def AkhirH = R25
.def AkhirL = R24
.def PointerH = R23
.def PointerL = R22
//register penghitung pengurangan//
.def Kurang2 = R21
.def Kurang1 = R20
//  register data sementara   //
.def Data4 = R19
.def Data3 = R18
.def Data2 = R17
.def Data1 = R16
//    register Tombol tekan   //
.def Tombol = R15
//register pembeda Sumber/Target//
.def SatauT = R14
//register pencatat ukuran file//
.def Ukur4 = R13
.def Ukur3 = R12
.def Ukur2 = R11
.def Ukur1 = R10
//register penamaan direktori//
.def Nama3 = R9
.def Nama2 = R8
.def Nama1 = R7
//register level Direktori//
.def sumber = R6
.def target = R5
.def cpy = R4
//register urutan file dalam direktori//
.def jmlurut= R3
.def nourut = R2
//=================================================================//
//                   Akhir deklarasi register                      //
//=================================================================//

/*#################################################################################################*/

//=================================================================//
//                   Alokasi alamat pada EEPROM                    //
//=================================================================//
//untuk merekam urutan file yang ditempuh
.equ jjksmbr= 0x0000
.equ jjktgt = 0x03B8
//jumlah file direktori total
.equ urut1 = 0x0770
//urutan file dalam direktori
.equ urut2 = 0x07B4
//untuk urutan direktori sumber
.equ urut3 = 0x07F8
//untuk urutan direktori target
.equ urut4 = 0x07F9
//untuk direktori sumber yang tak diolah
.equ urut5 = 0x07FA
//=================================================================//
//                   Akhir alokasi alamat                          //
//=================================================================//

/*#################################################################################################*/

//=================================================================//
//                      Inisialisasi UART                          //
//=================================================================//
.equ crystal = 16000000
.equ baud_rate = 2000000
.equ nilai_ubbr = (crystal / (8*baud_rate)) - 1
//=================================================================//
//                      Akhir inisialisasi                         //
//=================================================================//
//*****************************************************************//
//                      Akhir Deklarasi.asm                        //
//*****************************************************************//
```

```
//*****************************************************************//
//                        LCD Nokia 3310.asm                       //
//*****************************************************************//
.org 0x00
jmp utama
//=================================================================//
//                   Karakter ASCII untuk LCD                      //
//=================================================================//
;spasi
k1:
.db 0x00,0x00;
;!
k2:
.db 0x2F,0x00;
;#
k4:
.db 0x38,0x0F,0x3A,0x07,0x00,0x00;
;$
k5:
.db 0x4E,0xCB,0x52,0x32,0x00,0x00;
;%
k6:
.db 0x2F,0x19,0x3F,0x26,0x3D,0x00;
;&
k7:
.db 0x76,0x49,0x4B,0x7C,0x68,0x00;
;'
k8:
.db 0x07,0x00;
;(
k9:
.db 0x3E,0x41,0x00,0x00;
;)
k10:
.db 0x41,0x3E,0x00,0x00;
;+
k12:
.db 0x08,0x08,0x3E,0x08,0x08,0x00;
;,
k13:
.db 0xE0,0x00;
;-
k14:
.db 0x08,0x08,0x00,0x00;
;.
k15:
.db 0x20,0x00;
;0
k17:
.db 0x1E,0x21,0x21,0x1E,0x00,0x00;
;1
k18:
.db 0x02,0x3F,0x00,0x00;
;2
k19:
.db 0x23,0x31,0x2D,0x23,0x20,0x00;
;3
k20:
.db 0x21,0x21,0x25,0x1A,0x00,0x00;
;4
k21:
.db 0x08,0x0C,0x0A,0x3F,0x08,0x00;
;5
k22:
.db 0x27,0x25,0x19,0x00;
;6
k23:
.db 0x1C,0x26,0x25,0x18,0x00,0x00;
;7
k24:
.db 0x01,0x21,0x19,0x07,0x01,0x00;
;8
k25:
.db 0x1A,0x25,0x25,0x1A,0x00,0x00;
;9
k26:
.db 0x02,0x05,0x35,0x0E,0x00,0x00;
;;
k28:
.db 0xE2,0x00;
;=
k30:
.db 0x14,0x14,0x14,0x14,0x00,0x00;
;@
k33:
.db 0x3C,0x42,0xB5,0xAD,0xBD,0x22,0x1C,0x00;
;A
k34:
.db 0x30,0x1E,0x11,0x1E,0x30,0x00;
;B
k35:
.db 0x3F,0x25,0x25,0x1A,0x00,0x00;
;C
k36:
.db 0x1E,0x21,0x21,0x21,0x00,0x00;
;D
k37:
.db 0x3F,0x21,0x21,0x21,0x1E,0x00;
```

```
;E
k38:
.db 0x3F,0x25,0x25,0x21,0x00,0x00;
;F
k39:
.db 0x3F,0x05,0x05,0x01,0x00,0x00;
;G
k40:
.db 0x1E,0x21,0x21,0x25,0x3D,0x00;
;H
k41:
.db 0x3F,0x04,0x04,0x04,0x3F,0x00;
;I
k42:
.db 0x3F,0x00;
;J
k43:
.db 0x20,0x20,0x1F,0x00;
;K
k44:
.db 0x3F,0x0C,0x1A,0x21,0x00,0x00;
;L
k45:
.db 0x3F,0x20,0x20,0x00;
;M
k46:
.db 0x38,0x07,0x1C,0x20,0x1C,0x07,0x38,0x00;
;N
k47:
.db 0x3F,0x02,0x0C,0x10,0x3F,0x00;
;O
k48:
.db 0x1E,0x21,0x21,0x21,0x1E,0x00;
;P
k49:
.db 0x3F,0x09,0x09,0x06,0x00,0x00;
;Q
k50:
.db 0x1E,0x21,0x21,0x21,0x5E,0x40,0x00,0x00;
;R
k51:
.db 0x3F,0x09,0x19,0x26,0x00,0x00;
;S
k52:
.db 0x22,0x25,0x19,0x00;
;T
k53:
.db 0x01,0x01,0x3F,0x01,0x01,0x00;
;U
k54:
.db 0x1F,0x20,0x20,0x20,0x1F,0x00;
;V
k55:
.db 0x03,0x1C,0x20,0x1C,0x03,0x00;
;W
k56:
.db 0x1F,0x20,0x1C,0x03,0x1C,0x20,0x1F,0x00;
;X
k57:
.db 0x21,0x12,0x0C,0x12,0x21,0x00;
;Y
k58:
.db 0x01,0x02,0x3C,0x02,0x01,0x00;
;Z
k59:
.db 0x31,0x29,0x25,0x23,0x21,0x00;
;[
k60:
.db 0xFF,0x81,0x00,0x00;
;]
k62:
.db 0x81,0xFF,0x00,0x00;
;^
k63:
.db 0x06,0x01,0x06,0x00;
;_
k64:
.db 0x20,0x20,0x20,0x20,0x20,0x00;
;`
k65:
.db 0x01,0x02,0x00,0x00;
;a
k66:
.db 0x32,0x2A,0x2A,0x3C,0x00,0x00;
;b
k67:
.db 0x3F,0x22,0x22,0x1C,0x00,0x00;
;c
k68:
.db 0x1C,0x22,0x22,0x00;
;d
k69:
.db 0x1C,0x22,0x22,0x3F,0x00,0x00;
;e
k70:
.db 0x1C,0x2A,0x2A,0x2C,0x00,0x00;
;f
k71:
.db 0x04,0x3E,0x05,0x01,0x00,0x00;
;g
k72:
.db 0xAC,0xB2,0xB2,0x6E,0x00,0x00;
;h
k73:
.db 0x3F,0x02,0x02,0x3C,0x00,0x00;
;i
k74:
.db 0x04,0x3D,0x00,0x00;
```

```
;j
k75:
.db 0x84,0x7D,0x00,0x00;
;k
k76:
.db 0x3F,0x08,0x1C,0x22,0x00,0x00;
;l
k77:
.db 0x3F,0x00;
;m
k78:
.db 0x3E,0x02,0x02,0x3E,0x02,0x02,0x3C,0x00;
;n
k79:
.db 0x3E,0x02,0x02,0x3C,0x00,0x00;
;o
k80:
.db 0x1C,0x22,0x22,0x1C,0x00,0x00;
;p
k81:
.db 0xFE,0x22,0x22,0x1C,0x00,0x00;
;q
k82:
.db 0x1C,0x22,0x22,0xFE,0x00,0x00;
;r
k83:
.db 0x3E,0x02,0x02,0x00;
;s
k84:
.db 0x24,0x2A,0x12,0x00;
;t
k85:
.db 0x02,0x1F,0x22,0x00;
;u
k86:
.db 0x3E,0x20,0x20,0x3E,0x00,0x00;
;v
k87:
.db 0x02,0x1C,0x20,0x1C,0x02,0x00;
;w
k88:
.db 0x02,0x0C,0x30,0x0E,0x30,0x0C,0x02,0x00;
;x
k89:
.db 0x22,0x14,0x08,0x14,0x22,0x00;
;y
k90:
.db 0x06,0x98,0x78,0x06,0x00,0x00;
;z
k91:
.db 0x32,0x2A,0x26,0x22,0x00,0x00;
;{
k92:
.db 0x08,0xF7,0x81,0x00;
;}
k94:
.db 0x81,0xF7,0x08,0x00;
;~
k95:
.db 0x0C,0x04,0x08,0x08,0x0C,0x00;
//=============================================================================//
//                          Akhir kode tulisan                                 //
//=============================================================================//

/*#############################################################################*/
//=============================================================================//
//                          Tabel Konversi Tulisan                             //
//=============================================================================//
//  tabel konversi ASCII ke tampilan LCD   //
konversi:
.dW   k1,  k2,  k1,  k4,  k5,  k6,  k7,  k8;
.dW   k9,  k10, k1,  k12, k13, k14, k15, k1;
.dW   k17, k18, k19, k20, k21, k22, k23, k24;
.dW   k25, k26, k1,  k28, k1,  k30, k1,  k1;
.dW   k33, k34, k35, k36, k37, k38, k39, k40;
.dW   k41, k42, k43, k44, k45, k46, k47, k48;
.dW   k49, k50, k51, k52, k53, k54, k55, k56;
.dW   k57, k58, k59, k60, k1,  k62, k63, k64;
.dW   k65, k66, k67, k68, k69, k70, k71, k72;
.dW   k73, k74, k75, k76, k77, k78, k79, k80;
.dW   k81, k82, k83, k84, k85, k86, k87, k88;
.dW   k89, k90, k91, k92, k1,  k94, k95, k1;
//=============================================================================//
//                          Akhir Tabel Konversi                               //
//=============================================================================//
```

```
/*###########################################################################################*/

//=========================================================================================//
//                                      LCD Nokia 3310                                       //
//=========================================================================================//
//perintah LCD//
init_LCD:
.db 0x20,0x0c
;(PD=V=H=0)
;(D=1, E=0)
init_LCD2:
.db 0x21,0x13
;(PD=V=0, H=1)
;(BS2=0, BS1=BS0=0)
init_LCD3:
.db 0x06,0xC8
;(TC1=1, TC0=0)
;(Db5=DB4=DB2=DB1=Db0=0, Db6=DB3=1)
//perintah untuk menulis di baris pertama
LCD_baris1:
.db 0x81,0x41;
//perintah untuk menulis di baris kedua
LCD_baris2:
.db 0x80,0x42;
//perintah untuk menulis di baris ketiga
LCD_baris3:
.db 0x81,0x43;
//perintah untuk menulis di baris keempat
LCD_baris4:
.db 0x80,0x44;
//=========================================================================================//
//                                Akhir perintah LCD Nokia 3310                              //
//=========================================================================================//

/*###########################################################################################*/

//*****************************************************************************************//
//                                Pengiriman perintah ke LCD                                 //
//*****************************************************************************************//
prth_LCD:
  ldi     Data1,0x02
lpt1:
  lpm
  cbi     portb,4
  out     SPDR,r0
  call    cek_SPI
  adiw    Z,0x01
  dec     Data1
  cpi     Data1,0x00
  brne    lpt1
  out     SPDR,Data1
  call    cek_SPI
  ret
//*****************************************************************************************//
//                                Akhir pengririman perintah                                 //
//*****************************************************************************************//

/*###########################################################################################*/

//*****************************************************************************************//
//                                    Mengecek SPI Flag                                      //
//*****************************************************************************************//
cek_SPI:
  sbis    SPSR,SPIF
  jmp     cek_SPI
  ret
//*****************************************************************************************//
//                                Akhir pengecekan SPI Flag                                  //
//*****************************************************************************************//

/*###########################################################################################*/
```

```
//*****************************************************************************//
//                        Pengiriman Data ke LCD Nokia 3310                    //
//*****************************************************************************//
data_LCD:
   ldi    Data2,0x40
lpt2:
   push   Data2
   ldi    Data2,0x80
   cbi    portb,4
   out    SPDR,Data2
   call   cek_SPI
   pop    Data2
   out    SPDR,Data2
   call   cek_SPI
   ldi    Data1,0x54
lpt3:
   lpm
   sbi    portb,4
   ;kirim data
   out    SPDR,r0
   call   cek_SPI
   adiw   Z,0x01
   dec    Data1
   cpi    Data1,0x00
   brne   lpt3
   cbi    portb,4
   ;kirim perintah kosong
   out    SPDR,Data1
   call   cek_SPI
   inc    Data2
   cpi    Data2,0x46
   brne   lpt2
   ret
//*****************************************************************************//
//                    Akhir pengiriman Data ke LCD Nokia 3310                  //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                              Menghapus tampilan LCD                         //
//*****************************************************************************//
hapus:
   ldi    ZH,high(kosong*2)
   ldi    ZL,low(kosong*2)
   call   data_LCD
   ret
//*****************************************************************************//
//                          Akhir menghapus tampilan LCD                       //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                              Konversi Heksa ke Tulisan                      //
//*****************************************************************************//
Ubah:
   subi   Data1,0x20
   ldi    ZH,high(konversi*2)
   ldi    ZL,low(konversi*2)
lpt4:
   cpi    Data1,0x00
   breq   lpt5
   dec    Data1
   adiw   Z,0x02
   jmp    lpt4
lpt5:
   lpm    Data1,Z
   ldi    Data2,0x02
   mul    Data1,Data2
   movw   Z,R1:R0
lpt6:
   lpm    Data1,Z+
   sbi    portb,4
   out    SPDR,Data1
   call   cek_SPI
   cbi    portb,4
   cpi    Data1,0x00
   brne   lpt6
   ret
//*****************************************************************************//
//                        Akhir Konversi Heksa ke Tulisan                      //
//*****************************************************************************//
```

```
/*#########################################################################################*/

//*****************************************************************************************//
//                                  Menampilkan nama file                                  //
//*****************************************************************************************//
tpl_dt:
   call    LCD
   ldi     Data2,0x54
   ldi     Data1,0x00
lpt7:
   sbi     portb,4
   out     SPDR,Data1
   call    cek_SPI
   cbi     portb,4
   dec     Data2
   cpi     Data2,0x00
   brne    lpt7
   call    LCD
   ldi     YH,0x01
   ldi     YL,0x00
   ldi     Data2,0x00
lpt8:
   ld      Data1,Y+
   cpi     Data1,0x0D
   brne    lpt8
   inc     Data2
   cp      Data2,nourut
   brne    lpt8
lpt9:
   ld      Data1,Y+
   cpi     Data1,0x0D
   breq    lpt10
   call    Ubah
   rjmp    lpt9
lpt10:
   ret
//*****************************************************************************************//
//                                Akhir menampilkan nama file                              //
//*****************************************************************************************//

/*#########################################################################################*/

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//
//                              Data Tampilan LCD Nokia 3310                               //
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//

//=================================awal tampilan=========================================//
nama:
.db  0x00,0x00,0x80,0x70,0x4C,0x70,0x80,
     0x00,0xFC,0x00,0x90,0x50,0x50,0xE0,
     0x10,0xF8;
.db  0x10,0x00,0x00,0x00,0x00,0xFC,0x24,
     0x24,0x24,0x18,0x00,0xE0,0x50,0x50,
     0x60,0x00;
.db  0xF0,0x10,0x10,0xE0,0x00,0xE0,0x10,
     0x10,0xF0,0x00,0xFC,0x40,0x60,0x90,
     0x00,0xE0;
.db  0x10,0x10,0xE0,0x00,0xF0,0x10,0x10,
     0xE0,0x00,0xF4,0x00,0x00,0x00,0x00,
     0xFC,0x04;
.db  0x04,0x04,0x04,0xF8,0x00,0x90,0x50,
     0x50,0xE0,0x10,0xF8,0x10,0x00,0x90,
     0x50,0x50;
.db  0xE0,0x00,0x00,0x00,0x00,0x00,0x01,
     0x00,0x00,0x00,0x01,0x00,0x01,0x00,
     0x00,0x01;
.db  0x01,0xC1,0x00,0x00,0x01,0x00,0x00,
     0x00,0x00,0x01,0x80,0x00,0x00,0x00,
     0x00,0x00;
.db  0x01,0x01,0x01,0x00,0x01,0x00,0x00,
     0x01,0xC0,0x44,0x45,0x45,0x43,0x00,
     0xC1,0x00;
.db  0x00,0x00,0x01,0x00,0x01,0x01,0x00,
     0x00,0x07,0x01,0xC1,0x00,0x00,0x01,
     0x00,0x00;
.db  0x00,0x00,0x01,0x01,0x01,0xC1,0x01,
     0x00,0x00,0x00,0x41,0x01,0x01,0x00,
     0x00,0x01;
.db  0x00,0x00,0x01,0x01,0x01,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x18,0x07,0x04,0x07,
     0x18,0x00,0x1F,0x01,0x01,0x1E,0x01,
     0x0F,0x11;
.db  0x00,0x09,0x15,0x15,0x1E,0x00,0x1F,
     0x01,0x01,0x00,0x00,0x00,0x1F,0x02,
     0x02,0x02;
.db  0x02,0x00,0x1F,0x00,0x09,0x15,0x15,
     0x1E,0x00,0x12,0x15,0x15,0x09,0x00,
     0x1F,0x01;
```

```
.db  0x01,0x1E,0x00,0x04,0x04,0x04,0x0E,
     0x11,0x11,0x1F,0x00,0x1F,0x01,0x01,
     0x1F,0x00;
.db  0x01,0x0E,0x10,0x0E,0x01,0x0E,0x15,
     0x15,0x16,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0xFC,0x04,
     0x04,0x04,0x04,0xF8,0x00,0x00,0x70,
     0x80,0x70;
.db  0x80,0x70,0x00,0xF4,0x00,0x00,0x00,
     0x00,0xF8,0x04,0x04,0x04,0x04,0xF8,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x0C,0xF0,
     0x00,0xF0,0x0C,0xF0,0x00,0xF0,0x0C,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0xFC,0x08,0x30,
     0x40,0x80,0xFC,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x01,0x81,0x41,0x41,0x81,
     0x00,0x80;

//==================================Menu pemilihan port=========================//
plh_pt:
.db  0x00,0x00,0x00,0x00,0x00,0xF8,0x48,
     0x48,0x48,0x30,0x00,0xE8,0x00,0xF8,
     0x00,0xE8;
.db  0x00,0xF8,0x20,0x20,0xC0,0x00,0x00,
     0x00,0x00,0xF8,0x48,0x48,0x48,0x30,
     0x00,0xC0;
.db  0x20,0x20,0xC0,0x00,0xE0,0x20,0x20,
     0xF0,0x20,0x00,0x00,0x00,0x00,0x40,
     0xA0,0xA0;
.db  0x20,0x00,0xE0,0x00,0x00,0xE0,0x00,
     0xE0,0x20,0x20,0xC0,0x20,0x20,0xC0,
     0x00,0xF8;
.db  0x20,0x20,0xC0,0x00,0xC0,0xA0,0xA0,
     0xC0,0x00,0xE0,0x20,0x20,0x00,0x20,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x03,0x00,0x00,0x00,0x00,
     0x00,0x03;
.db  0x00,0x03,0x00,0x03,0x00,0x03,0x00,
     0x00,0x03,0x00,0x00,0x00,0x00,0x03,
     0x80,0x00;
.db  0x00,0x00,0x00,0x01,0x02,0x02,0x01,
     0x80,0x83,0x80,0x80,0x01,0x02,0x00,
     0x00,0x00;
.db  0x40,0x40,0x81,0x00,0x01,0x80,0xC0,
     0x01,0x00,0x00,0x00,0x00,0x80,0x41,
     0x41,0x81;
.db  0x01,0x80,0x40,0x40,0x81,0x00,0x80,
     0x40,0x40,0x80,0x00,0x01,0x00,0x00,
     0x80,0x41;
.db  0x40,0x80,0x00,0x81,0x40,0x40,0x80,
     0x00,0x81,0x40,0x40,0x80,0x00,0x81,
     0x40,0x40;
.db  0x81,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x0F,0x10;
.db  0x10,0x0F,0x00,0x18,0x14,0x12,0x11,
     0x00,0x01,0x00,0x1F,0x00,0x00,0x00,
     0x00,0x00;
.db  0x0F,0x10,0x10,0x0F,0x00,0x0F,0x12,
     0x12,0x0C,0x00,0x08,0x10,0x12,0x0D,
     0x00,0x00;
.db  0x00,0x00,0x0F,0x10,0x10,0x0F,0x00,
     0x0F,0x10,0x10,0x0F,0x00,0x08,0x10,
     0x12,0x0D;
.db  0x00,0x09,0x12,0x12,0x0F,0x00,0x00,
     0x00;

.db  0x00,0x02,0x02,0x02,0x01,0x00,0x01,
     0x02,0x02,0x03,0x00,0x03,0x00,0x80,
     0x03,0x00;
.db  0x00,0x03,0x00,0x03,0x02,0x02,0x01,
     0x00,0x01,0x02,0x02,0x02,0x00,0x03,
     0x00,0x00;
.db  0x00,0x02,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x02,0x01,0x3F,0x00,0x00,0x00,0x20,
     0x00,0x00,0x00,0x00,0x00,0x3F,0x04,0x04,
     0x04,0x03;
.db  0x00,0x1C,0x22,0x22,0x1C,0x00,0x3E,
     0x02,0x02,0x1F,0x22,0x00,0x00,0x00,
     0x00,0x02;
.db  0x01,0x3F,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x10,0x88,0x48,
     0x30,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0xF8;
.db  0x48,0x48,0x48,0x30,0x00,0xC0,0x20,
     0x20,0xC0,0x00,0xE0,0x20,0x20,0xF0,
     0x20,0x00;
.db  0x00,0x00,0x00,0x10,0x88,0x48,0x30,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x80,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x03,0x02,0x02,0x02,0x00,0x00,
     0x02,0x00;
.db  0x00,0x00,0x00,0x03,0x00,0x00,0x00,
     0x00,0x00,0x01,0x02,0x02,0x01,0x00,
     0x03,0x00;
.db  0x00,0x01,0x02,0x00,0x00,0x00,0x80,
     0x03,0x02,0x02,0x02,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x80,0x80;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x02,0x01,0x3F,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x1C,0x22,0x22,0x22,0x00,0x12,
     0x2A,0x2A,0x3C,0x00,0x3E,0x02,0x02,
     0x3C,0x00;
.db  0x1C,0x22,0x22,0x22,0x00,0x1C,0x2A,
     0x2A,0x2C,0x00,0x3F,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x31,0x28,0x24,0x23,0x00,0x00,
     0x00;

//========================Tulisan kosong untuk menghapus tampilan==================//
kosong:
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
. .db0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00;

//====================================Animasi1====================================//
gambar1:
.db 0x00,0x00,0x00,0x80,0xC0,0xE0,0xF0,
    0xF8,0xF8,0xFC,0xFC,0xFC,0xFC,0xFC,
    0xF8,0xF8;
.db 0xF0,0xF0,0xE0,0xC0,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x20,0x70,0xFE,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF;
.db 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF,0xFC,0xE0,0x40,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x01,0x87,0xFB,
    0xCF,0x7F;
.db 0xFF,0xFB,0xFF,0xFF,0xFF,0xFF,0x7F,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x80;
.db 0xEA,0x95,0xF6,0x5B,0xDF,0x7D,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFE,0xFF,0xFF,
    0xFF,0xFF;
.db 0xFF,0xFF,0xFF,0xFE,0xFC,0xFC,0x7C
    ,0xBC,0xF0,0xF0,0xE0,0xC0,0x80,0x80,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
.db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x80,0xC2,0xE2,0x5E,0xAB,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0x43,0xB7;
.db  0x4F,0x3F,0x0F,0xBF,0x4F,0xB7,0x4F,
     0x73,0x85,0xFB,0xFC,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,
     0xFE,0xFC,0xFC,0xF8,0xF0,0xE0,0x40,
     0x40,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0xF0,0xFC,0x7F,
     0xDF,0x25,0xDB,0xAC,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFF,0x55,0xAA,0x55,0x0A,0x31,
     0x06,0x01,0x8C,0xE1,0xFA,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0x7F,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00;

//==================================Animasi2==================================//
gambar2:
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x80,0xC0,0xE0,
     0xF0,0xF8;
.db  0xF8,0xFC,0xFC,0xFC,0xFC,0xFC,0xF8,
     0xF8,0xF0,0xF0,0xE0,0xC0,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x20,0x70,
     0xFE,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFC,0xE0,0x40,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x01,0x87,0xFB,0xCF,
     0x7F,0xFF,0xFB,0xFF,0xFF,0xFF,0xFF,
     0x7F,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x80,0xEA,0x95,0xF6,0x5B,0xDF,0x7D,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFE,0xFF,0xFF,0xFF
     ,0xFF,0xFF,0xFF,0xFF,0xFE,0xFC,0xFC,
     0x7C,0xBC;
.db  0xF0,0xF0,0xE0,0xC0,0x80,0x80,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x80,0xC2,0xE2,0x5E,
     0xAB,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x43,
     0xB7,0x4F,0x3F,0x0F,0xBF,0x4F,0xB7,
     0x4F,0x73;
.db  0x85,0xFB,0xFC,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFE,0xFE;
.db  0xFC,0xFC,0xF8,0xF0,0xE0,0x40,0x40,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
//==========================Animasi3==========================//
gambar3:
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x80,0xC0,0xE0,0xF0,
     0xF8,0xF8;
.db  0xFC,0xFC,0xFC,0xFC,0xFC,0xF8,0xF8,
     0xF0,0xF0,0xE0,0xC0,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x20,0x70,0xFE,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFC,0xE0,0x40,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x01,0x87,0xFB,0xCF,0x7F,
     0xFF,0xFB,0xFF,0xFF,0xFF,0xFF,0x7F,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x80,
     0xEA,0x95,0xF6,0x5B,0xDF,0x7D,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFE,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFE,0xFC,0xFC,0x7C,
     0xBC,0xF0;
.db  0xF0,0xE0,0xC0,0x80,0x80,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x40,0x40,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x80,0xC2,0xE2,0x5E,0xAB,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0x43,0xB7,
     0x4F,0x3F,0x0F,0xBF,0x4F,0xB7,0x4F,
     0x73,0x85;
.db  0xFB,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,
     0xFE,0xFC;
.db  0xFC,0xF8,0xF0,0xE0,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0xF0,0xFC,0x7F,
     0xDF,0x25;
.db  0xDB,0xAC,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0x55,0xAA,0x55,0x0A,0x31,
     0x06,0x01;
.db  0x8C,0xE1,0xFA,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0xFF,0xFF;
.db  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
     0x7F;
```

```
//=====================Tulisan untuk menunggu sejenak=======================//
tunggu:
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x38,0xA8;
.db  0x68,0xF8,0x48,0x40,0x40,0xE0,0x58,
     0x34,0x04,0x08,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x05,0x07,0x00,0x00,
     0x04,0x07,0x0D,0x00,0x06,0x05,0x07,
     0x04,0x05;
.db  0x03,0x01,0x06,0x05,0x1F,0x2C,0x32,
     0x1A,0x07,0x06,0x07,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x30,0x30,0x30,0x10,0xA0,0x60,
     0x50,0x70,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x03,0x05,
     0x04,0x02,0x01,0x00,0x07,0x04,0x06,
     0x05,0x05;
.db  0x07,0x03,0x27,0x35,0x26,0x1D,0x23,
     0x30,0x26,0x1D,0x03,0x00,0x00,0x07,
     0x04,0x06;
.db  0x05,0x04,0x00,0x04,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0xC0,0x20,0x20,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
```

```
.db  0x00,0x00,0x80,0x60,0x20,0x40,0x70,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x06,0x04,
     0x03,0x07,0x05,0x07,0x04,0x06,0x07,
     0x03,0x00;
.db  0x06,0x05,0x07,0x04,0x05,0x03,0x01,
     0x00,0x60,0x61,0x47,0x27,0x1C,0x06,
     0x07,0x05;
.db  0x07,0x04,0x06,0x05,0x05,0x07,0x03,
     0x07,0x07,0x04,0x00,0x04,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00;
```

//==========Tulisan akhir proses pengkopian==========//

selesai:

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0xC0,0xC0,0x40,0x40,0x00,
     0x80,0xC0,0xE0,0xA0,0xA0,0xE0,0xC0,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x80,0x80,0x80,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x01,0xE1,0xF1,0xF9,0x79,0x78,0x7B,
     0x8F,0xFE;
.db  0x70,0x00,0xF0,0xF8,0xA8,0xB8,0x30,
     0x80,0x80,0xF0,0x7E,0x1F,0x03,0xF0,
     0xF8,0xA8;
.db  0xB8,0x30,0xC0,0xC0,0xB0,0xF0,0x50,
     0x10,0xC0,0xE0,0xB0,0xD0,0xF0,0xB0,
     0x90,0x90;
.db  0xF0,0xF3,0xB3,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x01,0x01;
.db  0x01,0x01,0x01,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0xC0,0xC0,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0xC0,
     0xE0,0x20;
.db  0x20,0x20,0x30,0x30,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0xC0,0xC0,0x00,0x00,0x00,0x00,0x80,
     0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x34,0x3E,0x3F,
     0x15,0x04,0x3C,0x3E,0x2A,0x2E,0x0C,
     0x00,0x34;
.db  0x3C,0x0C,0x04,0x04,0x24,0x3C,0x3C,
     0x2C,0x00,0x24,0x34,0x3C,0x1C,0x3C,
     0x3C,0x0C;
.db  0x3C,0x3C,0x24,0x30,0x38,0x2C,0x34,
     0x3C,0x2C,0x24,0x00,0x00,0x00,0x00,
     0x20,0x38;
.db  0x1F,0x1F,0x3F,0x6E,0x46,0x70,0x38,
     0x2C,0x34,0x3C,0x2C,0x24,0x30,0x30,
     0x2C,0x3C;
.db  0x14,0x04,0x24,0x3C,0x3C,0x2C,0x00,
     0x00,0x30,0x3C,0x0F,0x37,0x3C,0x2C,
     0x00,0x00;
```

```
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00,0x00;
.db  0x00,0x00,0x00,0x00,0x00,0x00,0x00,
     0x00;


//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//
//                 Akhir Data Tampilan LCD Nokia 3310                  //
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//

//**************************************************************//
//                    Akhir LCD Nokia 3310.asm                  //
//**************************************************************//
```

```
//*********************************************************************//
//                          Memori.asm                                 //
//*********************************************************************//
//*********************************************************************//
//                  Merekam nama file kedalam SRAM                     //
//*********************************************************************//
rekam:
   call    setSRAM
   ldi     YH,0x01
   ldi     YL,0x00
   clr     jmlurut
//mengirim perintah untuk membuka direktori
   ldi     ZH,high(DIR*2)
   ldi     ZL,low(DIR*2)
   call    krm_UART
SRAM:
   call    trm_UART
//mengecek akhir nama file
   cpi     Data1,0x3E
   breq    SRAM1
   st      Y+,Data1
   rjmp    SRAM
SRAM1:
//terima 0x0D terakhir
   call    trm_UART
   ldi     YH,0x01
   ldi     YL,0x00
   clr     jmlurut
SRAM2:
//(hitung jumlah file) + 1
   ld      Data1,Y+
   cpi     Data1,0xff
   breq    SRAM3
   cpi     Data1,0x0D
   brne    SRAM2
   inc     jmlurut
   rjmp    SRAM2
   ldi     YH,0x01
   ldi     YL,0x00
SRAM3:
//cek ". DIR" dan ".. DIR"
   adiw    Y,0x01
   ld      Data1,Y
   ldi     Data2,0x2E
   cpse    Data1,Data2
   call    cekDirektori
   cpse    Data1,Data2
   rjmp    SRAM6
//cek apakah direktori adalah direktori kosong
   adiw    Y,0x0D
   ld      Data1,Y
   call    cekDirektori
SRAM4:
   sbiw    Y,0x0D
//set urutan file = 1
   clr     nourut
   inc     nourut
//kurangi jumlah (". DIR" dan ".. DIR")
   dec     jmlurut
   dec     jmlurut
   ldi     Data2,0xff
SRAM5:
//geser data dengan selang 13 data
   adiw    Y,0x0D
   ld      Data1,Y
//hapus data
   st      Y,Data2
   sbiw    Y,0x0D
   st      Y,Data1
   cpi     Data1,0x0D
   brne    SRAM5
   inc     nourut
//geser hingga data yang terakhir
   cp      nourut,jmlurut
   brne    SRAM5
SRAM6:
   ldi     Data1,0x01
   mov     nourut,Data1
   ret

//*********************************************************************//
//                  Akhir merekam nama file kedalam SRAM               //
//*********************************************************************//

/*#####################################################################*/
```

```
//*****************************************************************************//
//                              Setting isi SRAM                               //
//*****************************************************************************//
setSRAM:
  ldi   YH,0x01
  ldi   YL,0x00
SRAM7:
  ldi   Data1,0x0ff
  st    Y+,Data1
  cpi   YL,0xff
  brne  SRAM7
  cpi   YH,0x0C
  brne  SRAM7
  ret
//*****************************************************************************//
//                            Akhir setting isi SRAM                           //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Memasukkan data ke dalam SRAM                        //
//*****************************************************************************//
DATA:
  ldi   YH,0x10
  ldi   YL,0x00
  movw  Data3:Data4,AkhirL:AkhirH
SRAM8:
  call  trm_UART
  st    Y+,Data1
  dec   Data3
  cpi   Data3,0xff
  brne  SRAM8
  dec   Data4
  cpi   Data4,0xff
  brne  SRAM8
  call  prmpt
  ret
//*****************************************************************************//
//                      Akhir memasukkan data ke dalam SRAM                      //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                            Cek direktori file kosong                          //
//*****************************************************************************//
cekDirektori:
  ldi   Data2,0xff
  cpse  Data1,Data2
  ret
//jika iya, isi dengan data kosong (" = 0x22)
  ldi   Data1,0x22
  st    Y+,Data1
  ldi   Data1,0x0D
  st    Y,Data1
//karena ada penambahan nama file kosong
  inc   jmlurut
  sbiw  Y,0x01
  ret
//*****************************************************************************//
//                          Akhir cek direktori file kosong                      //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                              Cek ekstensi file                              //
//*****************************************************************************//
cekekstensi:
  ldi   YH,0x01
  ldi   YL,0x00
  mov   Data2,nourut
  inc   Data2
cek0D:
  ld    Data1,Y+
  cpi   Data1,0x0D
  brne  cek0D
  dec   Data2
  cpi   Data2,0x00
  brne  cek0D
```

```
//arahkan pointer ke posisi ekstensi
   sbiw   Y,0x02
   ld     Data1,Y
   cpi    Data1,0x52;cek R
   brne   nonDIR
   ld     Data1,-Y
   cpi    Data1,0x49;cek I
   brne   nonDIR
   ld     Data1,-Y
   cpi    Data1,0x44;cek D
   brne   nonDIR
//beri penanda akhir nama direktori
   ldi    Data1,0x0D
   st     -Y,Data1
   ldi    Data1,0x01
   rjmp   rekamposisi
nonDIR:
//penanda bukan file direktori
   ldi    Data1,0x02
rekamposisi:
//rekam posisi awal file
   movw pointerH:pointerL,X
   call   tulisEEPROM
   adiw   X,0x01
pointer_awal:
//arahkan ke pointer awal pembacaan nama
file
   ld     Data1,-Y
   cpi    Data1,0x0D
   brne   pointer_awal
   adiw   Y,0x01
salin_nama:
//salin nama file ke dalam EEPROM
   ld     Data1,Y+
   call   tulisEEPROM
   adiw   X,0x01
   cpi    Data1,0x0D
   brne   salin_nama
   ret
//*****************************************************************************//
//                          Akhir cek ekstensi file                            //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                              Baca isi EEPROM                                //
//*****************************************************************************//
bacaEEPROM:
   sbic   EECR,EEWE
   rjmp   bacaEEPROM
   out    EEARH,XH
   out    EEARL,XL
   sbi    EECR,EERE
   in     Data1,EEDR
   ret
//*****************************************************************************//
//                          Akhir baca isi EEPROM                              //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                              Tulis isi EEPROM                               //
//*****************************************************************************//
tulisEEPROM:
   sbic   EECR,EEWE
   rjmp   tulisEEPROM
   out    EEARH,XH
   out    EEARL,XL
   out    EEDR,Data1
   sbi    EECR,EEMWE
   sbi    EECR,EEWE
   ret
//*****************************************************************************//
//                          Akhir tulis isi EEPROM                             //
//*****************************************************************************//
//*****************************************************************************//
//                          Akhir Memori.asm                                   //
//*****************************************************************************//
```

```
//*****************************************************************//
//                        Penundaan.asm                            //
//*****************************************************************//
//*****************************************************************//
//                        Waktu Tunda                              //
//*****************************************************************//
Tunda:                                  NOP
   ldi     Data2,0x3f                   NOP
Tunda0xff:                              NOP
   ldi     Data1,0xff                   dec     Data1
TundaNOP:                               cpi     Data1,0x00
   NOP                                  brne    TundaNOP
   NOP                                  dec     Data2
   NOP                                  cpi     Data2,0x00
   NOP                                  brne    Tunda0xff
   NOP                                  ret
//*****************************************************************//
//                     Akhir Waktu Tunda                           //
//*****************************************************************//
//*****************************************************************//
//                     Akhir Penundaan.asm                         //
//*****************************************************************//
```

```
//*****************************************************************//
//                        Pilih_alamat.asm                         //
//*****************************************************************//
//*****************************************************************//
//              Pilih baris penulisan LCD (3 atau 4)               //
//*****************************************************************//
LCD:
   mov   Data1,SatauT
   cpi   Data1,0x00
   brne  LCDtarget
   ldi   ZH,high(LCD_baris2*2)
   ldi   ZL,low(LCD_baris2*2)
   rjmp  LCDtulis
LCDtarget:
   ldi   ZH,high(LCD_baris4*2)
   ldi   ZL,low(LCD_baris4*2)
LCDtulis:
   call  prth_LCD
   ret
//*****************************************************************//
//                Akhir pemilihan baris penulisan LCD              //
//*****************************************************************//

/*#################################################################################################*/

//*****************************************************************//
//              Pilih perintah pengaksesan Flash drive             //
//*****************************************************************//
Flash:
//cek kode file (1 = DIR; 2 = lainnya)
   call  bacaEEPROM
//arahkan pointer ke tempat awal nama file
   adiw  X,0x01
//rekam posisi
   movw pointerH:pointerL,X
//cek apakah file DIR?
   cpi   Data1,0x01
   brne  Flash2
//cek apakah sumber?
   mov   Data1,SatauT
   cpi   Data1,0x00
   breq  Flash1
MakeDisk:
   ldi   ZH,high(MKD*2)
   ldi   ZL,low(MKD*2)
   call  kirim_UART
   call  kirim_nama
   cpi   Data2,0x3E
   brne  MakeDisk
Flash1:
   movw X,pointerH:pointerL
ChangeDisk:
//buka file DIR
   ldi   ZH,high(CDFl*2)
   ldi   ZL,low(CDFl*2)
   call  kirim_UART
   call  kirim_nama
   cpi   Data2,0x3E
   brne  ChangeDisk
   ret
Flash2:
//bukan file DIR, cek target atau bukan
   mov   Data1,SatauT
   cpi   Data1,0x00
   brne  OpentoRead
   rjmp  OpentoWrite
OpentoRead:
   //untuk sumber
   //buka file untuk dibaca
   ldi   ZH,high(OPR*2)
   ldi   ZL,low(OPR*2)
   call  kirim_UART
   call  kirim_nama
   cpi   Data2,0x3E
   brne  OpentoRead
//rekam jumlah pengkopian
   mov   Data1,kurang1
   mov   Data2,kurang2
Flash3:
//arahkan pointer pembacaan file dalam disk
   cpi   Data1,0x00
   breq  Flash4
   ldi   ZH,high(SEK*2)
   ldi   ZL,low(SEK*2)
   call  krm_UART
   dec   Data1
//karena terpakai dalam prmpt
   push  Data2
   push  Data1
   call  prmpt
   pop   Data2
   pop   Data1
```

```
    rjmp    Flash3
Flash4:
//apakah selesai pencarian pointer?
    cpi     Data2,0xff
    breq    Flash5
    ldi     Data1,0xff
    dec     Data2
    rjmp    Flash3
Flash5:
//ukuran pembacaan (0x 0000(AkhirH)(AkhirL))
    ldi     ZH,high(RDF*2)
    ldi     ZL,low(RDF*2)
    call    kirim_UART
    ldi     Data1,0x00
    call    CTS
    sts     UDR1,Data1
    call    CTS
    sts     UDR1,Data1
    mov     Data1,AkhirH
    call    CTS
    sts     UDR1,Data1
    mov     Data1,AkhirL
    call    CTS
    sts     UDR1,Data1
    ldi     Data1,0x0D
    call    CTS
    sts     UDR1,Data1
//rekam hasil pembacaan file
    call    Data
    rjmp    CloseFile
OpentoWrite:
//untuk target
//buka file untuk ditulis
    ldi     ZH,high(OPW*2)
    ldi     ZL,low(OPW*2)
    call    kirim_UART
    call    kirim_nama
    cpi     Data2,0x3E
    brne    OpentoWrite
    ldi     YH,0x01
    ldi     YL,0x00
//memulai proses menulis file
    ldi     ZH,high(WRF*2)
    ldi     ZL,low(WRF*2)
    call    kirim_UART
    ldi     Data1,0x00
    call    CTS
    sts     UDR1,Data1
    call    CTS
    sts     UDR1,Data1
    mov     Data1,AkhirH
    call    CTS
    sts     UDR1,Data1
    mov     Data1,AkhirL
    call    CTS
    sts     UDR1,Data1
    movw    Data3:Data4,AkhirL:AkhirH
Flash6:
//cek hingga pembacaan yang terakhir
    ld      Data1,Y+
    call    CTS
    sts     UDR1,Data1
    dec     Data3
    cpi     Data3,0xff
    brne    Flash6
    dec     Data4
    cpi     Data4,0xff
    brne    Flash6
//respon prompt vinculum
    call    prmpt
//cek apakah ukuran data sebelumnya 0x0C00
    cpi     AkhirH,0x0C
    brne    CloseFile
    inc     Kurang1
//cek apakah penambahan LSB maksimum
    cpi     Kurang1,0x00
    brne    CloseFile
//tambahkan MSB,karena LSB maksimum
    inc     Kurang2
CloseFile:
//arahkan pointer ke awal nama file
    movw    X,pointerH:pointerL
//tutup file yang dibaca/ditulis
    ldi     ZH,high(CLF*2)
    ldi     ZL,low(CLF*2)
    call    kirim_UART
    call    kirim_nama
    cpi     Data2,0x3E
    brne    CloseFile
    ret
//*************************************************************************//
//                      Akhir pemilihan perintah                           //
//*************************************************************************//
//*************************************************************************//
//                      Akhir Pilih_alamat.asm                             //
//*************************************************************************//
```

```
//*****************************************************************//
//                          Proses.asm                             //
//*****************************************************************///
//*****************************************************************//
//                   Pencarian ditrektori file                     //
//*****************************************************************//
crDIR:
//rekam isi dari file direktori
   call    rekam
//rekam jumlah direktori(level pengkopian)
   ldi     Data1,0x00
   cp      SatauT,Data1
   breq    prss1
   clr     target
   rjmp    prss2
prss1:
   clr     sumber
prss2:
   call    tpl_dt
prss3:
   call    cek_tbl
   mov     Data2,Tombol
   cpi     Data2,0x01
   brne    prss4
//penanda batal operasi
   ldi     Data1,0xff
   ret
prss4:
//menuju ke file sebelumnya
   cpi     Data2,0x02
   brne    prss5
   call    Naik
   rjmp    prss3
prss5:
//mulai merekam file yang dikopi
   cpi     Data2,0x03
   brne    prss6
   call    Kopi
   ret
prss6:
// kembali ke isi dari direktori-parent
   cpi     Data2,0x04
   brne    prss7
   call    Kembali
   rjmp    prss3
prss7:
//melihat file berikutnya
   cpi     Data2,0x05
   brne    prss8
   call    Turun
   rjmp    prss3
prss8:
// menbuka isi direktori-child
   cpi     Data2,0x06
   brne    prss3
   call    Lanjut
   rjmp    prss3
//*****************************************************************//
//                Akhir pencarian direktori file                   //
//*****************************************************************//

/*#################################################################*/

//*****************************************************************//
//                   Menuju ke file sebelumnya                     //
//*****************************************************************//
Naik:
   mov     Data1,nourut
   cpi     Data1,0x01
   breq    prss9
   dec     nourut
prss9:
   call    tpl_dt
   ret
//*****************************************************************//
//              Akhir menuju ke file sebelumnya                    //
//*****************************************************************//

/*#################################################################*/
```

```
//*************************************************************************//
//                        Menuju ke file berikutnya                        //
//*************************************************************************//
Turun:
   mov   Data1,jmlurut
   dec   Data1
   cp    nourut,Data1
   breq  prss10
//*************************************************************************//
//                     Akhir menuju ke file berikutnya                     //
//*************************************************************************//

/*#########################################################################*/

//*************************************************************************//
//                        Membuka direktori - child                        //
//*************************************************************************//
Lanjut:
//cari file yang dipilih
   ldi   YH,0x01
   ldi   YL,0x00
   ldi   Data2,0x00
prss11:
   ld    Data1,Y+
   cpi   Data1,0x0D
   brne  prss11
   inc   Data2
   cp    Data2,nourut
   brne  prss11
   movw pointerH:pointerL,Y
   clr   Data3
prss12:
//arahkan pointer ke akhir pembacaan file
   ld    Data1,Y+
   cpi   Data1,0x0D
   brne  prss12
//pointer pada posisi setelah 0x0D
//cek apakah termasuk file direktori
   sbiw  Y,0x02
   ld    Data1,Y
   cpi   Data1,0x52;cek R
   brne  prss13
   ld    Data1,-Y
   cpi   Data1,0x49;cek I
   brne  prss13
   ld    Data1,-Y
   cpi   Data1,0x44;cek D
   breq  prss14
prss13:
//penanda file selain DIR
   ldi   Data1,0x02
   ret
prss14:
//tandai akhir nama
   inc   nourut
prss10:
   call  tpl_dt
   ret
   sbiw  Y,0x01
   ldi   Data1,0x0D
   st    Y,Data1
//proses merekam nomor urut file
   ldi   Data1,0x00
   cp    SatauT,Data1
   breq  prss15
//untuk target
   inc   target
   mov   Data1,target
   rjmp  prss16
prss15:
//untuk sumber
   inc   sumber
   ldi   XH,high(urut2)
   ldi   XL,low(urut2)
   mov   Data1,sumber
//tempatkan pointer urutan file
   dec   Data1
   add   XL,Data1
   push  Data1
   ldi   Data1,0x00
   adc   XH,Data1
   pop   Data1
//rekam nomor urut file
   mov   Data1,nourut
   call  tulisEEPROM
//rekam jumlah file dalam direktori
   ldi   XH,high(urut1)
   ldi   XL,low(urut1)
   add   XL,Data1
   ldi   Data1,0x00
   adc   XH,Data1
   mov   Data1,jmlurut
   call  tulisEEPROM
prss16:
//proses merekam path file
```

```
    ldi     Data1,0x00
    cp      SatauT,Data1
    breq    prss17
//untuk target
    ldi     XH,high(jjktgt)
    ldi     XL,low(jjktgt)
    mov     Data2,target
    rjmp    prss18
prss17:
//untuk sumber
    ldi     XH,high(jjksmbr)
    ldi     XL,low(jjksmbr)
    mov     Data2,sumber
prss18:
//scan level direktori
    cpi     Data2,0x01
    breq    prss20
prss19:
    call    bacaEEPROM
    adiw    X,0x01
    cpi     Data1,0x0D
    brne    prss19
    dec     Data2
    rjmp    prss18
prss20:
//*****************************************************************************//
//                          Akhir membuka direktori - child                    //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Membuka kembali direktori – parent                 //
//*****************************************************************************//
Kembali:
    ldi     ZH,high(CD*2)
    ldi     ZL,low(CD*2)
    call    krm_UART
    call    prmpt
    call    rekam
    call    tpl_dt
    ldi     Data2,0x02
prss22:
    call    bacaEEPROM
    cpi     Data1,0x0D
    brne    prss23
    dec     Data2
    cpi     Data2,0x00
    breq    prss24
prss23:
    ldi     Data1,0xff
    call    tulisEEPROM
    sbiw    X,0x01
    jmp     prss22
//rekam dan buka file direktori
    movw Y,pointerH:pointerL
//penanda file DIR
    ldi     Data1,0x01
    call    tulisEEPROM
    adiw    X,0x01
    ldi     ZH,high(CDFl*2)
    ldi     ZL,low(CDFl*2)
    call    kirim_UART
prss21:
    ld      Data1,Y+
//kirim ke VNC1L-1A
    call    CTS
    sts     UDR1,Data1
//tulis ke EEPROM
    call    tulisEEPROM
    adiw    X,0x01
    cpi     Data1,0x0D
    brne    prss21
    call    prmpt
    call    rekam
    call    tpl_dt
//penanda file DIR
    ldi     Data1,0x01
    ret
prss24:
    ldi     Data1,0x00
    cp      SatauT,Data1
    breq    prss25
    cpse    target,Data1
    dec     target
    ret
prss25:
    cpse    sumber,Data1
    dec     sumber
    ldi     XH,high(urut2)
    ldi     XL,low(urut2)
//tempatkan pointer urutan file
    add     XL,sumber
    ldi     Data1,0x00
    adc     XH,Data1
//rekam nomor urut file
    ldi     Data1,0xff
    call    tulisEEPROM
//rekam jumlah file dalam direktori
```

```
   ldi     XH,high(urut1)
   ldi     XL,low(urut1)
   add     XL,sumber
   ldi     Data1,0x00
   adc     XH,Data1
   ldi     Data1,0xff
   call    tulisEEPROM
   ret
//*****************************************************************************//
//                          Akhir membuka direktori – parent                   //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Mulai mengunci file                                //
//*****************************************************************************//
Kopi:
//cek apakah file DIR?, bila ya rekam
   call    lanjut
   cpi     Data1,0x02
   brne    prss27
//bila ya, apakah sumber?
   mov     Data1,SatauT
   cpi     Data1,0x00
   brne    prss27
//rekam nama file
   movw Y,pointerH:pointerL
   ldi     Data1,0x02
   call    tulisEEPROM
   adiw    X,0x01
prss26:
   ld      Data1,Y+
   call    tulisEEPROM
   adiw    X,0x01
   cpi     Data1,0x0D
   brne    prss26
   inc     sumber
   mov     cpy,sumber
prss27:
//cek apakah target
   ldi     Data1,0x01
   cpse    Data1,SatauT
   rjmp    prss31
//buat direktori baru
   ldi     Data1,0x30
   mov     nama1,Data1
   mov     nama2,Data1
   mov     nama3,Data1
prss28:
   ldi     ZH,high(MKD*2)
   ldi     ZL,low(MKD*2)
   call    kirim_UART
   call    CTS
   sts     UDR1,nama1
   call    CTS
   sts     UDR1,nama2
   call    CTS
   sts     UDR1,nama3
   ldi     Data1,0x0D
   call    CTS
   sts     UDR1,Data1
//terima respon
   call    prmpt
//cek apakah "Comand Failed"
   cpi     Data2,0x43
   breq    prss29
//cek apakah "Disk Full"
   cpi     Data2,0x44
   brne    prss30
//penanda disk penuh
   ldi     Data1,0xfe
   ret
prss29:
   inc     nama3
   mov     Data1,nama3
   cpi     Data1,0x3A
   brne    prss28
   ldi     Data1,0x30
   mov     nama3,Data1
   inc     nama2
   mov     Data1,nama2
   cpi     Data1,0x3A
   brne    prss28
   ldi     Data1,0x30
   mov     nama2,Data1
   inc     nama1
   mov     Data1,nama1
   cpi     Data1,0x3A
   brne    prss28
   ldi     Data1,0xfe
   ret
prss30:
//rekam direktori file
   ldi     Data1,0x01
   call    tulisEEPROM
   adiw    X,0x01
   mov     Data1,nama1
   call    tulisEEPROM
```

```
    adiw   X,0x01
    mov    Data1,nama2
    call   tulisEEPROM
    adiw   X,0x01
    mov    Data1,nama3
    call   tulisEEPROM
    adiw   X,0x01
    ldi    Data1,0x0D
    call   tulisEEPROM
    inc    target
    ldi    XH,high(urut4)
    ldi    XL,low(urut4)
    mov    Data1,target
    call   tulisEEPROM
    rjmp   prss32
prss31:
    ldi    XH,high(urut3)
    ldi    XL,low(urut3)
    mov    Data1,sumber
    call   tulisEEPROM
    ldi    XH,high(urut5)
    ldi    XL,low(urut5)
    mov    Data1,cpy
    call   tulisEEPROM
prss32:
//penanda siap melakukan pengkopian
    ldi    Data1,0xfc
    ret
//*************************************************************************//
//                        Akhir mengunci file                              //
//*************************************************************************//

/*#########################################################################*/

//*************************************************************************//
//                     Pengurangan ukuran file                             //
//*************************************************************************//
kurang:
    ldi    akhirH,0x0C
    cp     ukur3,akhirH
    brlo   prss33
    rjmp   prss35
prss33:
    ldi    Data1,0x00
    cp     ukur2,Data1
    brne   prss34
    cp     ukur1,Data1
    breq   prss36
    dec    ukur1
prss34:
    dec    ukur2
    sub    AkhirH,ukur3
    clr    ukur3
prss35:
    sub    ukur3,AkhirH
    ldi    AkhirH,0x0C
    clr    AkhirL
    ret
prss36:
    mov    AkhirH,ukur3
    clr    ukur3
    mov    AkhirL,ukur4
    clr    ukur4
    ret
//*************************************************************************//
//                  Akhir Pengurangan ukuran file                          //
//*************************************************************************//
//*************************************************************************//
//                        Akhir Proses.asm                                 //
//*************************************************************************//
```

```
//*****************************************************************//
//                          Tombol.asm                             //
//*****************************************************************//
//*****************************************************************//
//                      Pengecekan tombol                          //
//*****************************************************************//
cek_tbl:
   cbi     portb,6
   sbi     porte,7
   ldi     Data2,0x01
   ldi     Data1,0x01
   call    TundaNOP
//tombol1
   sbic    pind,5
   rjmp    tombol2
   ldi     Data1,0x01
   mov     Tombol,Data1
tombol1_angkat:
   sbis    pind,5
   rjmp    tombol1_angkat
   ret
tombol2:
   sbic    pind,6
   rjmp    tombol3
   ldi     Data1,0x02
   mov     Tombol,Data1
tombol2_angkat:
   sbis    pind,6
   rjmp    tombol2_angkat
   ret
tombol3:
   sbic    pind,7
   rjmp    tombol4
   ldi     Data1,0x03
   mov     Tombol,Data1
tombol3_angkat:
   sbis    pind,7
   rjmp    tombol3_angkat
   ret
tombol4:
   sbi     portb,6
   cbi     porte,7
   ldi     Data2,0x01
   ldi     Data1,0x01
   call    TundaNOP
   sbic    pind,5
   rjmp    tombol5
   ldi     Data1,0x04
   mov     Tombol,Data1
tombol4_angkat:
   sbis    pind,5
   rjmp    tombol4_angkat
   ret
tombol5:
   sbic    pind,6
   rjmp    tombol6
   ldi     Data1,0x05
   mov     Tombol,Data1
tombol5_angkat:
   sbis    pind,6
   rjmp    tombol5_angkat
   ret
tombol6:
   sbic    pind,7
   rjmp    cek_tbl
   ldi     Data1,0x06
   mov     Tombol,Data1
tombol6_angkat:
   sbis    pind,7
   rjmp    tombol6_angkat
   ret
//*****************************************************************//
//                Akhir pengecekan penekanan tombol                //
//*****************************************************************//
//*****************************************************************//
//                        Akhir Tombol.asm                         //
//*****************************************************************//
```

```
//*****************************************************************//
//                          VNC1L–1A.asm                           //
//*****************************************************************//
//=================================================================//
//                       Vinculum  VNC1L-1A                        //
//=================================================================//
; Melihat isi direktori
DIR:
.db 0x01,0x0D;
; Melihat ukuran file
DIR_file:
.db 0x01,0x20;
;Mengembalikan ke direktori sebelumnya
CD:
.db 0x02,0x20,0x2E,0x2E,0x0D,0x00;
; Membuka direktori
CDF1:
.db 0x02,0x20;
; Membuat DIR
MKD:
.db 0x06,0x20;
; ukuran penulisan
WRF:
.db 0x08,0x20;
; buka file untuk tulis
OPW:
.db 0x09,0x20;
; Menutup file
CLF:
.db 0x0A,0x20;
; ukuran pembacaan
RDF:
.db 0x0B,0x20;
; Membuka file untuk dibaca
OPR:
.db 0x0E,0x20;
;Mengarahkan pointer pembacaan
SEK:
.db 0x28,0x20,0x00,0x00,0x0C,0x00,0x0D,0x00;
; Melihat sisa ruang dalam Flash drive
FS:
.db 0x93,0x0D;
//=================================================================//
//                    Akhir perintah VNC1L-1A                      //
//=================================================================//

/*#################################################################################*/

//*********************************************************************//
//                      Cek carriage return (CR) (0x0D)                //
//*********************************************************************//
Enter:                                  brne   Enter
   call    trm_UART                     ret
   cpi     Data1,0x0D
//*********************************************************************//
//                    Akhir carriage return (CR) (0x0D)                //
//*********************************************************************//

/*#################################################################################*/

//*********************************************************************//
//                              Cek prompt                             //
//*********************************************************************//
prmpt:                                  call    Enter
   call    trm_UART                     ret
   mov     Data2,Data1
//*********************************************************************//
//                          Akhir cek prompt                           //
//*********************************************************************//

/*#################################################################################*/
```

```
//*****************************************************************************//
//                          Terima data dari VNC1L-1A                          //
//*****************************************************************************//
trm_UART:
; RTS (low)
  cbi     portD,4
cek_rx:
  lds     Data1,UCSR1A
  sbrs    Data1,RXC1
  jmp     cek_rx
  sbi     portD,4
  lds     Data1,UDR1
  ret
//*****************************************************************************//
//                      Akhir penerimaan data dari VNC1L-1A                    //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                      Mengecek flash drive sudah tertancap                   //
//*****************************************************************************//
cek_port:
  sbic    portb,0
  rjmp    ubah_port
  sbi     portb,0
  rjmp    cek_port0
ubah_port:
  cbi     portb,0
cek_port0:
  call    prmpt
  cpi     Data2,0x3E
  brne    cek_port0
cek_port1:
  call    Tunda
  call    prmpt
  cpi     Data2,0x3E
  brne    cek_port1
Cek3E:
  call    CTS
  ldi     Data1,0x0D
  sts     UDR1,Data1
  call    prmpt
  cpi     Data2,0x3E
  brne    Cek3E
  ret
//*****************************************************************************//
//                  Akhir pengecekan penancapan flash drive                    //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Cek kesiapan VNC1L-1A                              //
//*****************************************************************************//
CTS:
  sbic    pinD,1
  rjmp    CTS
cek_tx:
  lds     Data2,UCSR1A
  sbrs    Data2,UDRE1
  jmp     cek_tx
  ret
//*****************************************************************************//
//                      Akhir cek kesiapan VNC1L-1A                            //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Kirim perintah utuh ke Vinculum                    //
//*****************************************************************************//
krm_UART:
  lpm     Data1,Z+
  call    CTS
  sts     UDR1,Data1
  cpi     Data1,0x0D
  brne    krm_UART
  ret
//*****************************************************************************//
//                      Akhir kirim perintah utuh ke Vinculum                  //
//*****************************************************************************//
```

```
/*#############################################################################*/

//*****************************************************************************//
//                          Kirim perintah tak utuh ke Vinculum                //
//*****************************************************************************//
kirim_UART:                                     cpi     Data1,0x20
   lpm     Data1,Z+                             brne    kirim_UART
   call    CTS                                  ret
   sts     UDR1,Data1
//*****************************************************************************//
//                      Akhir kirim perintah tak utuh ke Vinculum              //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Baca dan kirim nama file ke Vinculum               //
//*****************************************************************************//
kirim_nama:                                     cpi     Data1,0x0D
   call    bacaEEPROM                           brne    kirim_nama
   call    CTS                                  call    prmpt
   sts     UDR1,Data1                           ret
   adiw    X,0x01
//*****************************************************************************//
//                      Akhir baca dan kirim nama file ke Vinculum             //
//*****************************************************************************//

/*#############################################################################*/

//*****************************************************************************//
//                          Cek flashdrive apakah sudah dicabut                //
//*****************************************************************************//
cabut:                                          mov     Data2,Data1
   call    CTS                                  call    prmpt
   ldi     Data1,0x0D                           cpi     Data2,0x4E
   sts     UDR1,Data1                           brne    cabut
//cek respon ND0x0D ((0x4E)(0x44)0x0D)          ret
   call    trm_UART
//*****************************************************************************//
//                      Akhir Cek flashdrive apakah sudah dicabut              //
//*****************************************************************************//
//*****************************************************************************//
//                          Akhir VNC1L–1A.asm                                 //
//*****************************************************************************//
```

Datasheet

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 64K Bytes of In-System Reprogrammable Flash Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
  - 2K Bytes EEPROM Endurance: 100,000 Write/Erase Cycles
  - 4K Bytes Internal SRAM
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the JTAG Standard
- Extensive On-chip Debug Support
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 1 to 16 Bits
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain (1x, 10x, 200x)
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega64L
  - 4.5 - 5.5V for ATmega64
- Speed Grades
  - 0 - 8 MHz for ATmega64L
  - 0 - 16 MHz for ATmega64

ATmega64:

8-bit Microcontroller with 64K Bytes In-System Programmable Flash

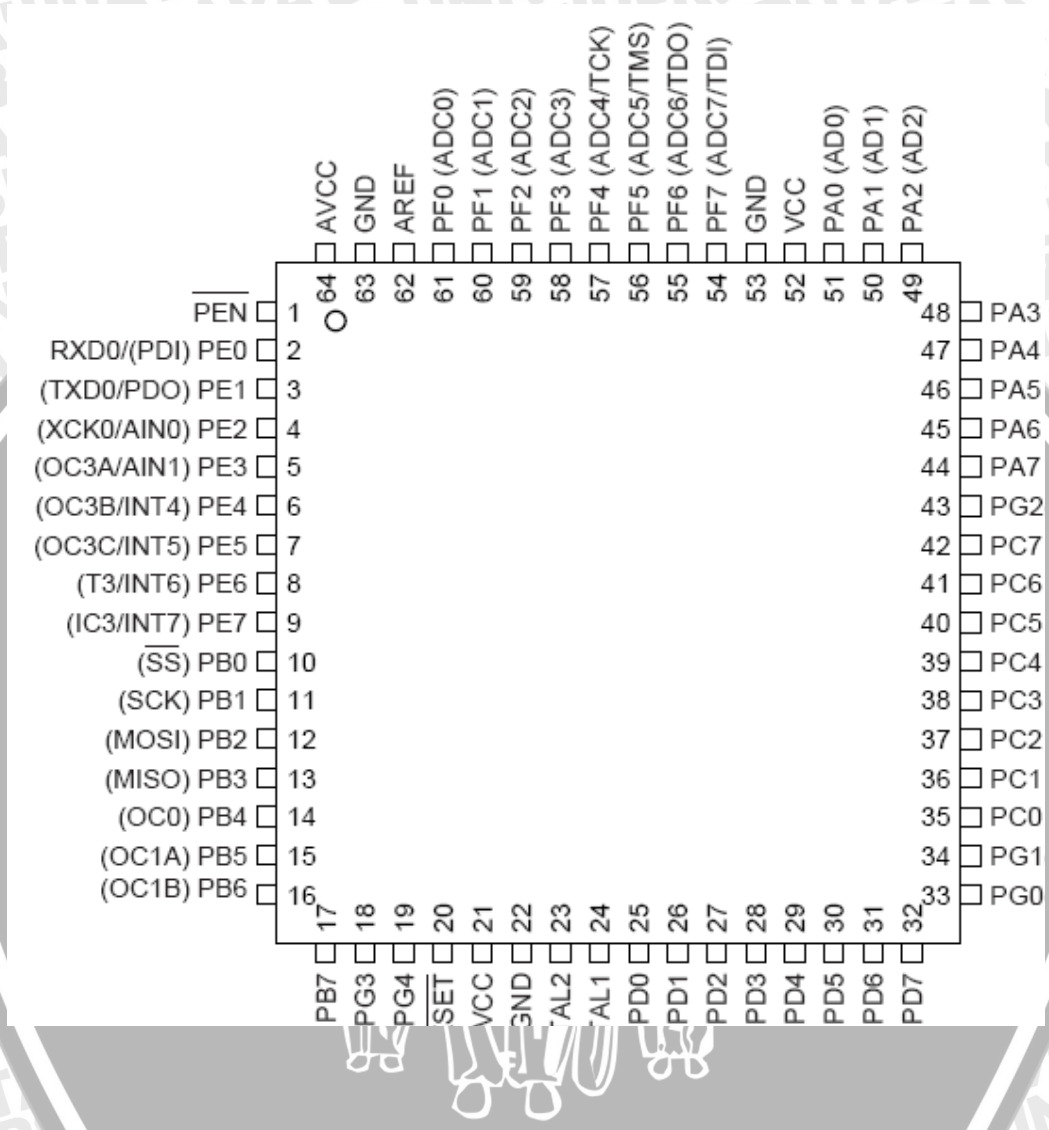
ATmega64
ATmega64L
Preliminary

Rev. 2490D–AVR–02/03

Pin Configuration

Figure 1. Pinout ATmega64

TQFP/MLF



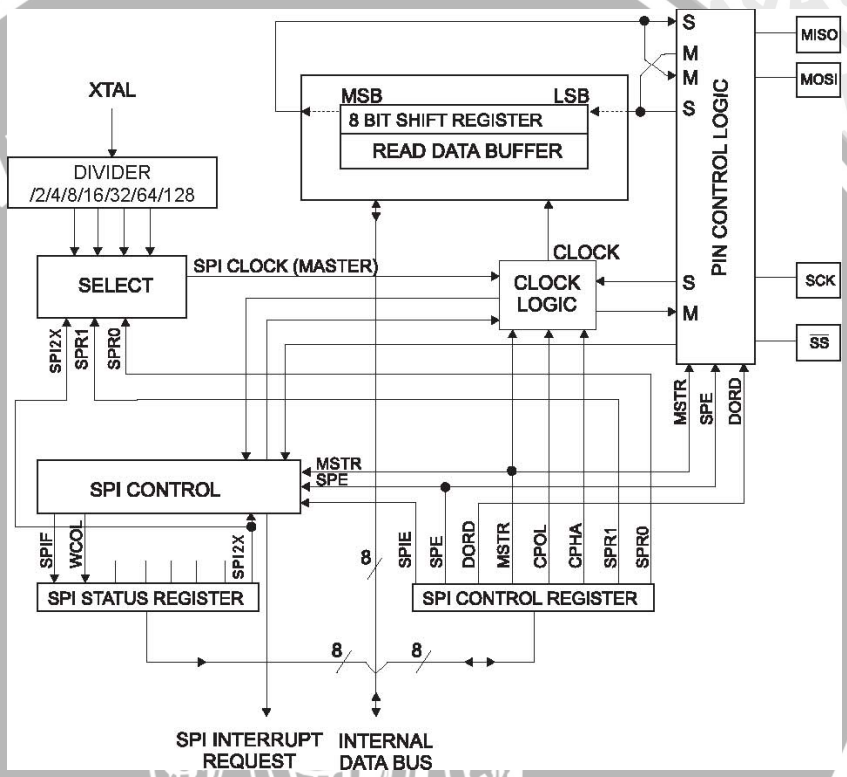
Disclaimer Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega64 and peripheral devices or between several AVR devices. The ATmega64 SPI includes the following features:

- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

Figure 75. SPI Block Diagram[1]



Note: 1. Refer to Figure 1 on page 2, and Table 30 on page 71 for SPI pin placement.

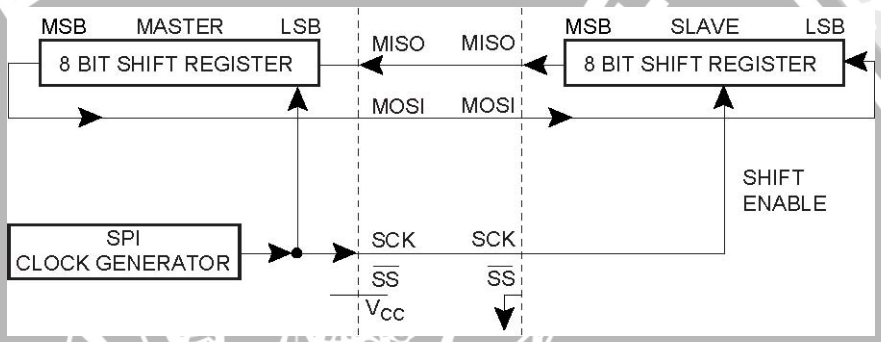
The interconnection between Master and Slave CPUs with SPI is shown in Figure 76. The system consists of two Shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select SS pin of the desired Slave. Master and Slave prepare the data to be sent in their respective Shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, SS, line.

When configured as a Master, the SPI interface has no automatic control of the SS line. This must be handled by user software before communication can start. When this is

done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, SS line. The last incoming byte will be kept in the buffer register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the SS pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the SS pin is driven low. As one byte has been completely shifted, the end of transmission flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the buffer register for later use.

Figure 76. SPI Master-Slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{osc}/4$

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and SS pins is overridden according to Table 69. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 68.

Table 69. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
SS	User Defined	Input

Note: 1. See “Alternate Functions of Port B” on page 71 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example, if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.

Assembly Code Example(1)

```
SPI_MasterInit:
; Set MOSI and SCK output, all others input
ldi r17,(1<<DD_MOSI)|(1<<DD_SCK)
out DDR_SPI,r17
; Enable SPI, Master, set clock rate fck/16
ldi r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
out SPCR,r17
ret
SPI_MasterTransmit:
; Start transmission of data (r16)
out SPDR,r16
Wait_Transmit:
; Wait for transmission complete
sbis SPSR,SPIF
rjmp Wait_Transmit
ret
```

C Code Example(1)

```
void SPI_MasterInit(void)
{
/* Set MOSI and SCK output, all others input */
DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
/* Enable SPI, Master, set clock rate fck/16 */
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
void SPI_MasterTransmit(char cData)
{
/* Start transmission */
 SPDR = cData;
/* Wait for transmission complete */
 while(!(SPSR & (1<<SPIF))) ;
}
```

Note: 1. The example code assumes that the part specific header file is included.

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example(1)

```
SPI_SlaveInit:
; Set MISO output, all others input
ldi r17,(1<<DD_MISO)
out DDR_SPI,r17
; Enable SPI
ldi r17,(1<<SPE)
out SPCR,r17
ret
SPI_SlaveReceive:
; Wait for reception complete
sbis SPSR,SPIF
rjmp SPI_SlaveReceive
; Read received data and return
in r16,SPDR
ret
```

C Code Example(1)

```
void SPI_SlaveInit(void)
{
/* Set MISO output, all others input */
DDR_SPI = (1<<DD_MISO);
/* Enable SPI */
SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
/* Wait for reception complete */
while(!(SPSR & (1<<SPIF))) ;
/* Return data register */
return SPDR;
}
```

Note: 1. The example code assumes that the part specific header file is included.

SS Pin Functionality

Slave Mode

When the SPI is configured as a Slave, the Slave Select (SS) pin is always input. When SS is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When SS is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the SS pin is driven high.

The SS pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the Master clock generator. When the SS pin is driven high, the SPI Slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the SS pin.

If SS is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the SS pin of the SPI Slave.

If SS is configured as an input, it must be held high to ensure Master SPI operation. If the SS pin is driven low by peripheral circuitry when the SPI is configured as a Master with the SS pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that SS is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – SPIE: SPI Interrupt Enable

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the Global Interrupt Enable bit in SREG is set.

• Bit 6 – SPE: SPI Enable

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

• Bit 5 – DORD: Data Order

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

• Bit 4 – MSTR: Master/Slave Select

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

• Bit 3 – CPOL: Clock Polarity

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 77 and Figure 78 for an example. The CPOL functionality is summarized below:

Table 70. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

• Bit 2 – CPHA: Clock Phase

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 77 and Figure 78 for an example. The CPHA functionality is summarized below:

Table 71. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

• Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in Table 72.

Table 72. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If SS is an input and is driven low when the SPI is in Master mode, this will also set the SPIF flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

• Bit 6 – WCOL: Write COLlision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

• Bit 5..1 – Res: Reserved Bits

These bits are reserved bits in the ATmega64 and will always read as zero.

• Bit 0 – SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 72). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at f/4 or lower.

osc

The SPI interface on the ATmega64 is also used for program memory and EEPROM downloading or uploading. See page 305 for SPI Serial Programming and verification.

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

Data Modes There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 77 and Figure 78. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 70 and Table 71, as done below:

Table 73. CPOL and CPHA Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

Figure 77. SPI Transfer Format with CPHA = 0

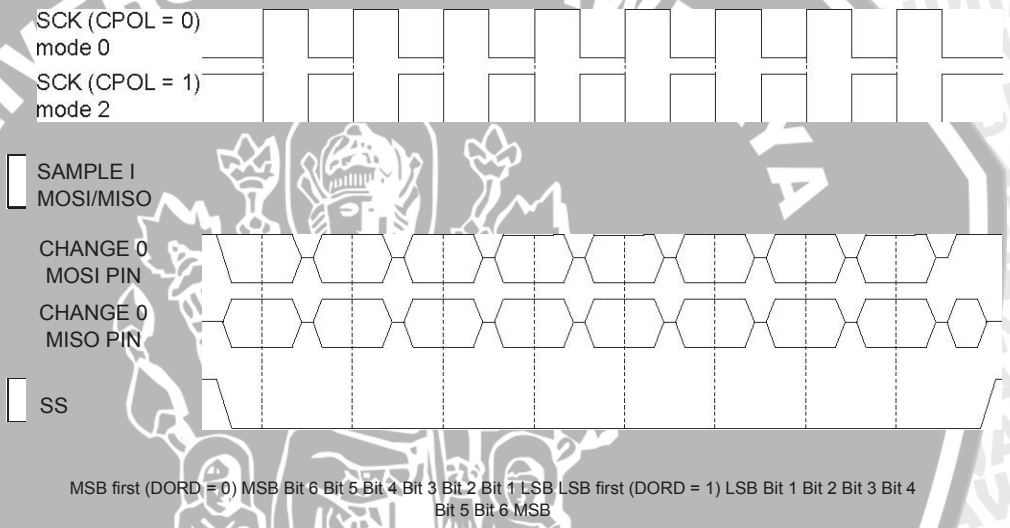
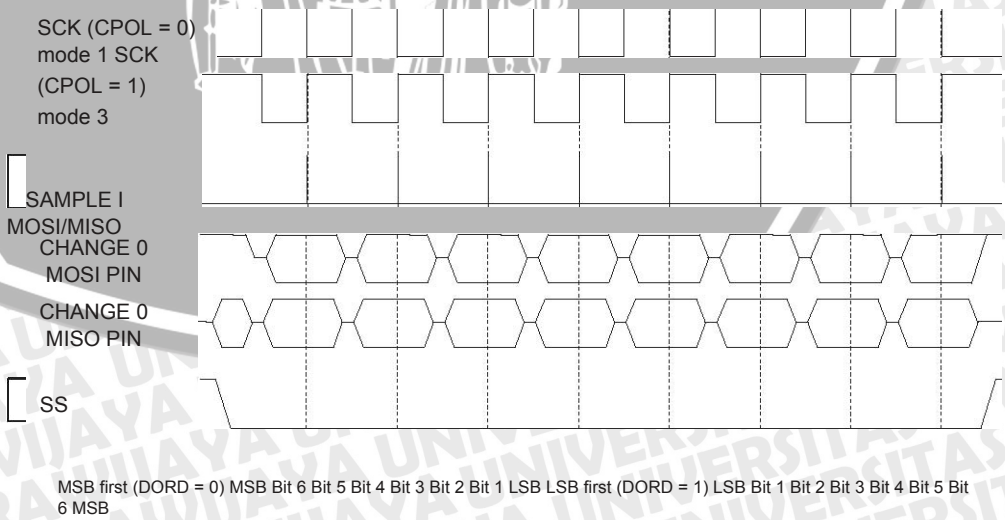


Figure 78. SPI Transfer Format with CPHA = 1



USART

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Master or Slave Clocked Synchronous Operation**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

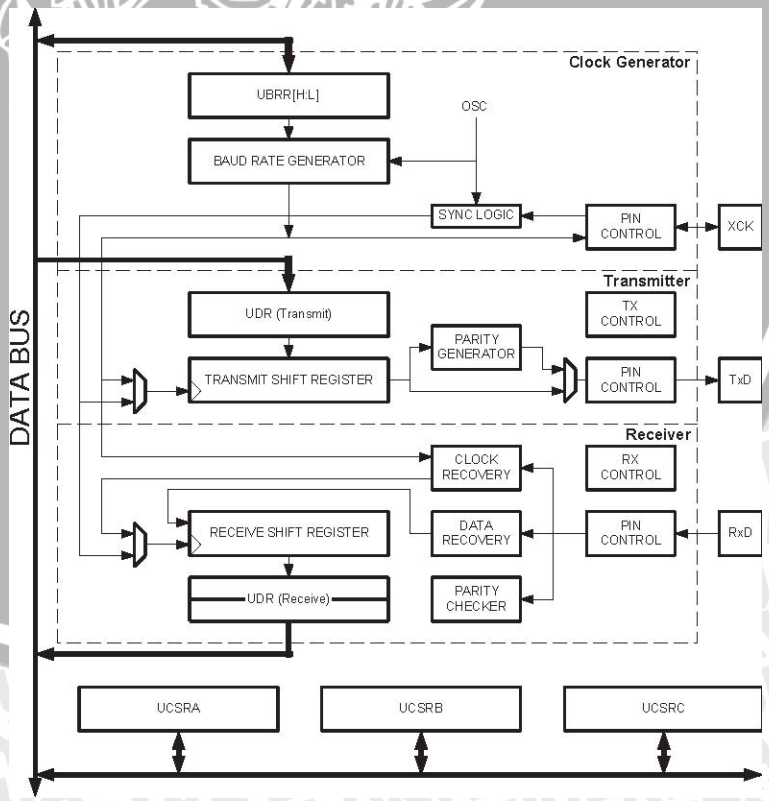
Dual USART

The ATmega64 has two USART's, USART0 and USART1. The functionality for both USART's is described below. USART0 and USART1 have different I/O Registers as shown in "Register Summary" on page 339. Note that in ATmega103 compatibility mode, USART1 is not available, neither is the UBRR0H or UCRS0C registers. This means that in ATmega103 compatibility mode, the ATmega64 supports asynchronous operation of USART0 only.

Overview

A simplified block diagram of the USART Transmitter is shown in Figure 79. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 79. USART Block Diagram (1)



Note: 1. Refer to Figure 1 on page 2, Table 36 on page 75, and Table 39 on page 78 for USART pin placement.

USART Register Description

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE flag in the UCSRA Register is set. Data written to UDR when the UDRE flag is not set, will be ignored by the USART transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

• Bit 7 – RXC: USART Receive Complete

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

• Bit 6 – TXC: USART Transmit Complete

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

• Bit 5 – UDRE: USART Data Register Empty

The UDRE flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the Transmitter is ready.

• Bit 4 – FE: Frame Error

This bit is set if the next character in the receive buffer had a Frame Error when received. For example, when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

• Bit 3 – DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

• Bit 2 – UPE: USART Parity Error

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

• Bit 1 – U2X: Double the USART Transmission Speed

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

• Bit 0 – MPCM: Multi-processor Communication Mode

This bit enables the Multi-processor Communication Mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 184.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – RXCIE: RX Complete Interrupt Enable

Writing this bit to one enables interrupt on the RXC flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

• Bit 6 – TXCIE: TX Complete Interrupt Enable

Writing this bit to one enables interrupt on the TXC flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

• Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable

Writing this bit to one enables interrupt on the UDRE flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

• Bit 4 – RXEN: Receiver Enable

Writing this bit to one enables the USART receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and UPE flags.

• Bit 3 – TXEN: Transmitter Enable

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

• Bit 2 – UCSZ2: Character Size
The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

• Bit 1 – RXB8: Receive Data Bit 8
RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

• Bit 0 – TXB8: Transmit Data Bit 8
TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC(1)

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Note: 1. This register is not available in ATmega103 compatibility mode.

• Bit 7 – Reserved Bit
This bit is reserved for future use. For compatibility with future devices, these bit must be written to zero when UCSRC is written.

• Bit 6 – UMSEL: USART Mode Select
This bit selects between asynchronous and synchronous mode of operation.

Table 77. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

• Bit 5:4 – UPM1:0: Parity Mode
These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the UPE flag in UCSRB will be set.

Table 78. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

• Bit 3 – USBS: Stop Bit Select
This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 79. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

• **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 80. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• **Bit 0 – UCPOL: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 81. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

USART Baud Rate Registers – UBRRL and UBRRH (1)

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Note: 1. UBRRH is not available in mega103 compatibility mode

• **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

• **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 82 to Table 85. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see “Asynchronous Operational Range” on page 183). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature..... -55°C to +125°C
Storage Temperature ..... -65°C to +150°C
Voltage on any Pin except RESET
with respect to Ground .....-1.0V to V_{CC}+0.5V
Voltage on RESET with respect to Ground.....-1.0V to +13.0V
Maximum Operating Voltage 6.0V
DC Current per I/O Pin ..... 40.0 mA
DC Current V_{CC} and GND Pins..... 200.0 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	Except XTAL1 and RESET pins	-0.5		0.2 V_{CC}(1)	V
V_{IL1}	Input Low Voltage	XTAL1 pin, External Clock Selected	-0.5		0.1 V_{CC} (1)	V
V_{IL2}	Input Low Voltage	RESET pin	-0.5		0.2 V_{CC} (1)	V
V_{IH}	Input High Voltage	Except XTAL1 and RESET pins	0.6 V_{CC} (2)		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	XTAL1 pin, External Clock Selected	0.7 V_{CC} (2)		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	RESET pin	0.85 V_{CC} (2)		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(3) (Ports A,B,C,D, E, F, G)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	Output High Voltage(4) (Ports A,B,C,D)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.0 2.2			V V
I_{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			8.0	µA
I_{IH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			8.0	µA
R_{RST}	Reset Pull-up Resistor		30		100	kΩ
R_{PEN}	PEN Pull-up Resistor		25		100	kΩ
R_{PU}	I/O Pin Pull-up Resistor		33		122	kΩ

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units
		Active 4 MHz, VCC = 3V (ATmega64L)			5	mA
		Active 8 MHz, VCC = 5V (ATmega64)			20	mA
ICC	Power Supply Current	Idle 4 MHz, VCC = 3V (ATmega64L)			2	mA
		Idle 8 MHz, VCC = 5V (ATmega64)			12	mA
	Power-down mode(5)	WDT enabled, VCC = 3V		< 25	40	µA
		WDT disabled, VCC = 3V		< 10	25	µA
VACIO	Analog Comparator Input Offset Voltage	VCC = 5V Vin = VCC/2			40	mV
IACLK	Analog Comparator Input Leakage Current	VCC = 5V Vin = VCC/2	-50		50	nA

Notes: 1. “Max” means the highest value where the pin is guaranteed to be read as low

1 “Min” means the lowest value where the pin is guaranteed to be read as high

2 Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed: TQFP and MLF Package: 1] The sum of all IOL, for all ports, should not exceed 400 mA. 2] The sum of all IOL, for ports A0 - A7, G2, C3 - C7 should not exceed 300 mA. 3] The sum of all IOL, for ports C0 - C2, G0 - G1, D0 - D7, XTAL2 should not exceed 150 mA. 4] The sum of all IOL, for ports B0 - B7, G3 - G4, E0 - E7 should not exceed 150 mA. 5] The sum of all IOL, for ports F0 - F7, should not exceed 200 mA. If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.

3 Although each I/O port can source more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: TQFP and MLF Package: 1] The sum of all IOH, for all ports, should not exceed 400 mA. 2] The sum of all IOH, for ports A0 - A7, G2, C3 - C7 should not exceed 300 mA. 3] The sum of all IOH, for ports C0 - C2, G0 - G1, D0 - D7, XTAL2 should not exceed 150 mA. 4] The sum of all IOH, for ports B0 - B7, G3 - G4, E0 - E7 should not exceed 150 mA. 5] The sum of all IOH, for ports F0 - F7, should not exceed 200 mA. If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

4 Minimum V_{CC} for Power-down is 2.5V.

SPI Timing Characteristics

Table 135. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 72		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		TBD		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t_{sck}		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		ns
10	SCK period	Slave	4 • t_{ck}			
11	SCK high/low(1)	Slave	2 • t_{ck}			
12	Rise/Fall time	Slave		TBD		
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		
16	SCK to SS high	Slave	20			
17	SS high to tri-state	Slave		10		
18	SS low to SCK	Slave	20			

Note: 1. In SPI Programming mode the minimum SCK high/low period is:
-2 t_{CLCL} for f_{ck} < 12 MHz
-3 t_{CLCL} for f_{ck} >12 MHz

Figure 157. SPI Interface Timing Requirements (Master Mode)

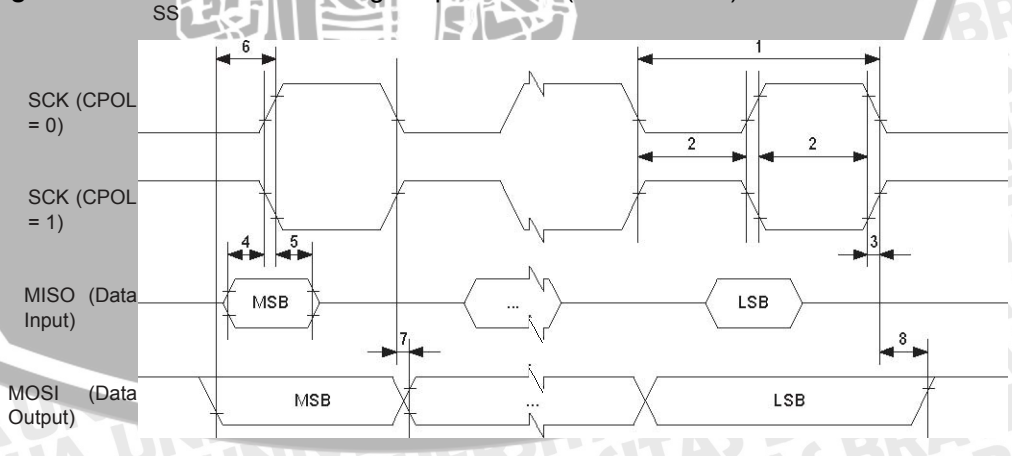


Figure 158. SPI Interface Timing Requirements (Slave Mode)