SKRIPSI

PROGRAMMABLE TIMER PERALATAN LISTRIK YANG DIAKSES VIA SMS (SHORT MESSAGE SERVICE)

Disusun Oleh :

RANU SETYOBUDI

NIM : 0017062

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
MARET 2009

LEMBAR PERSETUJUAN

PROGRAMMABLE TIMER PERALATAN LISTRIK YANG DIAKSES VIA SMS (SHORT MESSAGE SERVICE)

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat
Guna Mencapai Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :
RANU SETYOBUDI
NIM : 0017062

Diperiksa dan Disetujui,
Dosen Pembimbing,

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

Mengetahui ,
Ketua Jurusan Teknik Elektro

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
STRATA

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2009

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNIK INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl.Raya Karanglo Km. 2
Malang

BERITA ACARA UJIAN SKRIPSI

Nama	:	Ranu Setyobudi
NIM	:	00.17.062
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika
Judul Skripsi	:	Programmable Timer Peralatan Listrik Yang Diakses Via SMS (Short Message Service)

Dipertahankan di hadapan Majelis Penguji Skripsi jenjang Strata Satu (S-1) pada:

Hari	: Senin
Tanggal	: 23 Maret 2009
Dengan Nilai	: 82,75 (A)

Panitia Majelis Penguji :

Ketua

Ir. Sidik Noertjahjono, MT
NIP. Y. 1028700167

Sekretaris

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

Anggota Penguji :

Penguji Pertama

Ir. Mimien Mustikawati, MT
NIP. P. 1030000352

Penguji Kedua

I Komang Somawirata, ST. MT
NIP. P.1030100361

PREFAZIONE

ALLA PRIMA EDIZIONE

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNIK INDUSTRI
JURUSAN TEKNIK ELEKTRO
Jl.Raya Karanglo Km. 2
Malang

FORMULIR PERBAIKAN UJIAN SKRIPSI

Dalam melaksanakan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi Teknik Elektronika, perlu adanya perbaikan skripsi untuk mahasiswa :

Nama	: Ranu Setyobudi
NIM	: 00.17.062
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika
Masa Bimbingan	: 24 Oktober 2008 – 24 April 2009
Judul skripsi	: Programmable Timer Peralatan Listrik Yang Diakses Via SMS (Short Message Service)

No.	Tanggal	Uraian	Paraf
1.	23 Maret 2009	• Abstraksi Bahasa Inggris Dihilangkan	

Mengetahui,

Dosen Penguji I

Ir. Mimien Mustikawati, MT
NIP.P. 1030000352

Dosen Penguji II

I Komang Somawirata, ST, MT
NIP.P 1030100361

Dosen Pembimbing

Ir. F. Yudi Limpraptono, MT
NIP.Y. 1039500274

Abstract

ABSTRAKSI

PROGRAMMABLE TIMER PERALATAN LISTRIK
YANG DIAKSES VIA SMS (SHORT MESSAGE SERVICE)

(Ranu Setyobudi, NIM 0017062, T. ELEKTRONIKA S-1)
(Dosen Pembimbing : Ir. F. Yudi Limpraptono, MT)

Abstrak

Selain kebutuhan sarana dan prasarana komunikasi, pengendalian peralatan listrik sudah menjadi kebutuhan pokok bagi manusia. Salah satu contohnya adalah untuk menyalakan timer peralatan listrik dari jarak jauh pada saat pemilik rumah sedang sibuk berada diluar rumah atau berpergian.

Alat ini menggunakan mikrokontroler ATMega8535 sebagai pengendali utama keseluruhan rangkaian. Handphone digunakan untuk menerima SMS, LCD digunakan sebagai display, relay digunakan sebagai penghubung peralatan listrik yang dikontrol dengan tegangan jala-jala PLN AC220 V.

Alat ini dapat bekerja dengan baik mengontrol peralatan listrik dalam dua mode yaitu baik mode tanpa timer maupun pada mode menggunakan timer. Sistem sms sangat tergantung pada keadaan jaringan, dari penerimaan sinyal pada handphone user maupun pada handphone base terminal.

Kata Kunci : ATMega8535, SMS, LCD, Relay.

APPENDIX

KATA PENGANTAR

Puji dan syukur penyusun panjatkan kepada Allah SWT yang telah melimpahkan rahmat dan petunjuk-Nya sehingga penulis dapat menyelesaikan laporan skripsi dengan judul :

" PROGRAMMABLE TIMER PERALATAN LISTRIK YANG DIAKSES VIA SMS (SHORT MESSAGE SERVICE) "

Pembuatan skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang S-1 di Institut Teknologi Nasional Malang. Laporan skripsi ini merupakan tanggung jawab penulis atas ilmu pengetahuan yang didapat selama mengikuti perkuliahan.

Atas terselesaikannya skripsi ini, penulis mengucapkan terima kasih kepada :

1. Bapak DR. Ir. Abraham Lomi, MSEE selaku rektor ITN Malang.
2. Bapak Ir. F. Yudi Limpraptono, MT, selaku dosen pembimbing dan Ketua Jurusan Teknik Elektro ITN Malang yang telah memberikan bimbingan, pengarahan serta ilmu yang sangat berharga sehingga skripsi ini dapat terselesaikan.
3. Keluarga yang telah memberikan dukungan demi terselesaikannya skripsi ini.
4. Teman-teman yang telah membantu menyelesaikan skripsi ini.

Menyadari adanya keterbatasan pengetahuan, referensi, dan pengalaman, Penyusun mengharapkan saran dan masukan demi lebih baiknya skripsi ini.

Semoga skripsi ini dapat bermanfaat bagi penyusun khususnya dan bagi para pembaca umumnya.

Malang, Maret 2009

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring dengan berkembangnya teknologi nirkabel (wireless), salah satunya adalah teknologi GSM (Global System for Mobile Communications), yang semakin murah dan dengan kapasitas jangkauan yang semakin luas, menyebabkan pemakaian telepon seluler tidak hanya berada pada salah satu golongan masyarakat tertentu saja (kaum elit), namun pemakai telepon seluler sudah menjangkau semua lapisan. SMS (Short Message Service) adalah salah satu fasilitas yang terdapat pada telepon seluler yang hampir setiap orang mengenalnya. Selain memiliki biaya operasional yang cukup murah, fasilitas ini juga merupakan media komunikasi dan sarana informasi antar individu yang cukup memiliki sifat waktu yang nyata (real-time), sehingga tidaklah mengherankan apabila SMS masih tetap menjadi pilihan bagi setiap orang sebagai sarana komunikasi.

Selain kebutuhan sarana dan prasarana komunikasi, pengendalian peralatan listrik sudah menjadi kebutuhan pokok bagi manusia. Salah satu contohnya adalah untuk menyalakan timer peralatan listrik dari jarak jauh pada saat pemilik rumah sedang sibuk berada diluar rumah atau berpergian. Tingginya aktivitas manusia dalam usahanya memenuhi kebutuhan ekonomi semakin mengurangi waktu untuk berada di sekitar rumah, baik untuk berkumpul bersama keluarga maupun untuk melakukan kegiatan-kegiatan yang lainnya. Permasalahan

ini mendorong manusia untuk mengadakan perancangan peralatan pengendali yang efektif dan efisien. Berangkat dari permasalahan tersebut maka penulis merancang **"Programmable Timer Peralatan Listrik diakses Via SMS"**.

Alat ini dirancang untuk memberikan kemudahan-kemudahan kepada kita agar dapat mengendalikan berbagai peralatan listrik dari jarak jauh sesuai seting timer yang kita inginkan menggunakan SMS. Sebagai contoh penggunaan alat ini adalah dapat digunakan untuk mengaktifkan proses pengisian air pada bak mandi, menyalakan heater, mengaktifkan alat pengharum ruangan, mengaktifkan proses penyiraman taman bunga dll, sesuai dengan seting timer yang kita inginkan via SMS. Dengan menggunakan SMS maka diharapkan biaya operasional penggunaan alat ini menjadi murah dengan memberikan manfaat yang dapat diandalkan.

1.2. Rumusan Masalah

Dari uraian tersebut diatas maka timbul beberapa permasalahan diantaranya adalah sebagai berikut:

1. Perancangan hardware meliputi:
   - Antarmuka komunikasi serial Mikrokontroler dengan Handphone.
   - Antarmuka Mikrokontroler dengan Rangkaian Driver Relay sebagai penghubung peralatan listrik yang di kontrol dengan tegangan jala-jala dari PLN AC220V.
   - Antarmuka Mikrokontroler dengan LCD M1632 sebagai display timer dan kondisi peralatan listrik ON/OFF.

2. Perancangan software (program) untuk mikrokontroler dengan menggunakan bahasa Assembly.
3. Melakukan pengujian per blok rangkaian melalui pengukuran tegangan pada input dan output tiap-tiap rangkaian.
4. Melakukan pengujian secara keseluruhan dengan cara meng-inputkan seting timer peralatan listrik melalui perintah teks via sms.
5. Memasukkan hasil pengujian ke dalam tabel dan melakukan analisis data.

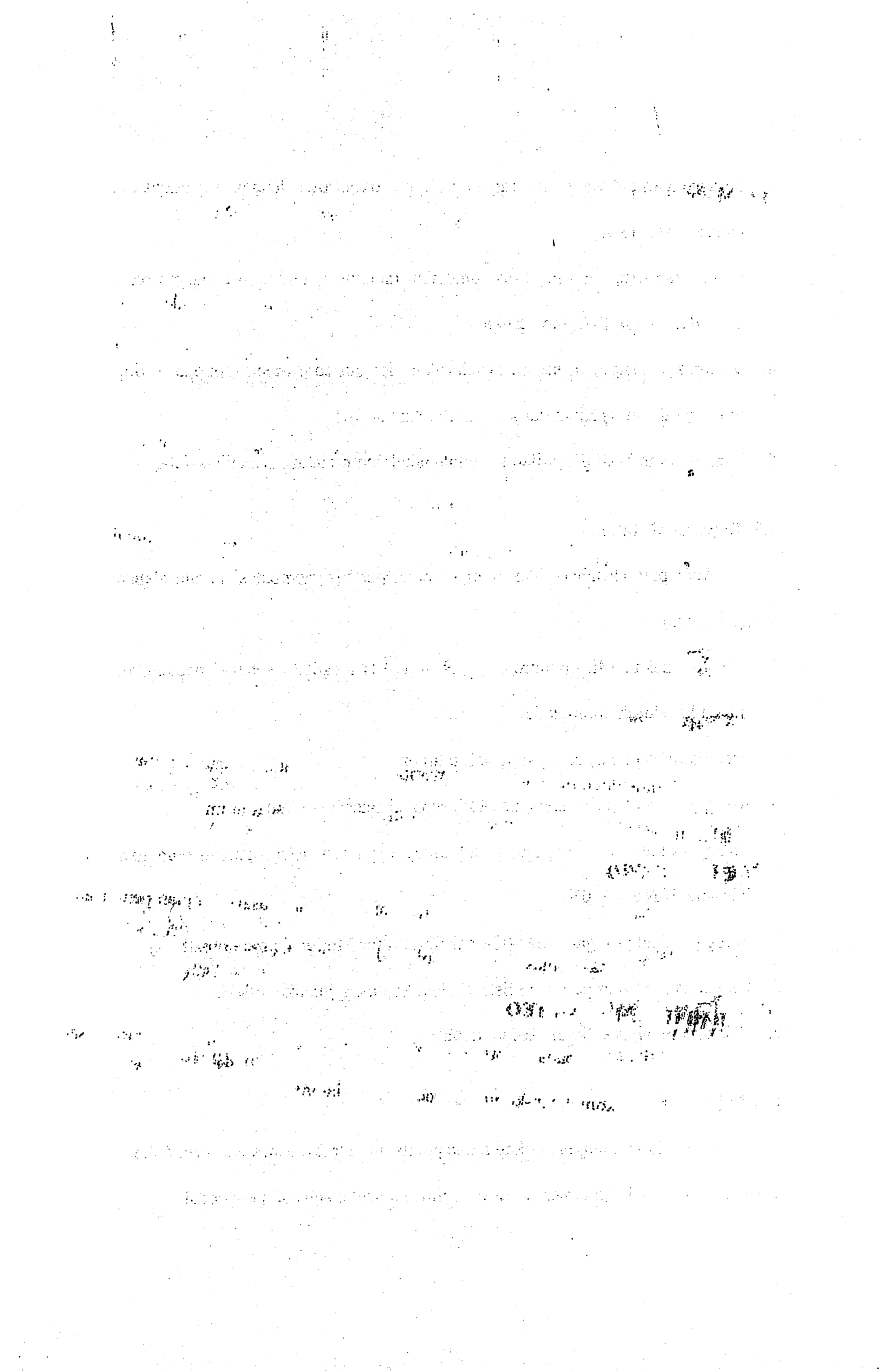
1.3. Batasan Masalah

Agar permasalahan tidak meluas maka penulis membatasi permasalahan sebagai berikut:

1. Mengggunakan Mikrokontroler AVR seri ATMega8535 sebagai pengendali utama keseluruhan rangkaian.
2. Menggunakan delay sebagai pewaktu timer.
3. Menggunakan Hp tipe siemens M35i sebagai Base Terminal.
4. Menggunakan Relay sebagai penghubung peralatan listrik dengan tegangan jala-jala PLN AC220V.
5. Menggunakan jaringan GSM (Global System for Mobile Communications).
6. Pengendalian timer peralatan listrik dilakukan menggunakan SMS.
7. Tidak membahas software secara detail.

1.4. Tujuan

Untuk dapat mengendalikan *timer* peralatan listrik diakses via SMS (Short Message Service) dengan menggunakan mikrokontroler sebagai pengendali.



1.5. Metodologi

Dengan pembuatan skripsi ini menggunakan metodologi sebagai berikut :

- Study Literatur.
  - Pada studi literatur akan dipelajari tentang media komunikasi data serial antara hp dan mikrokontroler, cara penggunaanya dan cara pembuatan aplikasinya.
  - Mempelajari tentang sistem pengendalian timer peralatan listrik, cara penggunaannya dan cara pembuatan aplikasinya.
  - Mempelajari pemrograman mikrokontroler dengan menggunakan bahasa Assembly.
- Melakukan studi analisa.

1.6. Sistematika Penulisan

Untuk mempermudah dan memperjelas pembahasan dari laporan tugas akhir ini penulis menerapkan system penulisan seperti dibawah ini :

BAB I : PENDAHULUAN

Tentang latar belakang masalah, rumusan masalah, tujuan penulisan, batasan masalah, metodologi dan sistematika penulisan.

BAB II : DASAR TEORI

Membahas teori dasar rangkaian yang digunakan dan teori dasar komponen-komponen pendukung lainnya.

BAB III : PERENCANAAN SISTEM

Perancangan Programmable Timer Peralatan Listrik diakses Via SMS.

BAB IV : PENGUJIAN SISTEM

Pengujian Programmable Timer Peralatan Listrik diakses Via SMS yang telah dibuat, secara per blok rangkaian maupun secara keseluruhan.

BAB V : PENUTUP

Kesimpulan pembahasan pada bab–bab sebelumnya dan saran untuk pengembangan alat.

BAB II

DASAR TEORI

2.1. Umum

Pengiriman pesan menggunakan teknologi seluler dipercaya menjadi *trend* terbaru dan dapat diaplikasikan untuk berbagai kebutuhan. Telepon seluler dengan fasilitas SMS mampu bertukar informasi berbasis teks secara jarak jauh (remote) dan tanpa kabel (wireless) sehingga dapat memberikan solusi yang tepat terhadap masalah pengontrolan timer peralatan listrik secara jarak jauh. Ditambah dengan dukungan teknologi mikrokontroler yang memungkinkan dibentuknya sebuah sistem yang memiliki efisiensi daya dan tempat, menjadikan telepon seluler sebagai sarana alternatif selain sebagai sarana komunikasi juga dapat dijadikan sebagai sarana pengendali jarak jauh.

Penggunaan mikrokontroler dalam berbagai aplikasi memang memberikan banyak keuntungan tapi juga tidak luput dari kekurangan. Keuntungan yang dapat diperoleh dengan menggunakan mikrokontroler antara lain: banyak pilihan yang ditawarkan tergantung kebutuhan, murah, bisa digunakan untuk bermacam-macam aplikasi, berdaya rendah, dan hanya memerlukan sedikit tambahan komponen luar dan proses penanganannya yang mudah baik dari segi operasi maupun aplikasinya. Sedangkan kekurangan dari mikrokontroler adalah keterbatasan memori didalamnya sehingga tidak mampu menangani program-program yang cukup besar dan rumit.

2.2. SMS (Short Message Service)

SMS merupakan salah satu layanan pesan teks yang dikembangkan dan distandarisasi oleh suatu badan yang bernama ETSI (European Telecomunication Standards Institute) sebagai bagian dari pengembangan GSM Phase 2, yang terdapat pada dokumentasi GSM 03.40 dan GSM 03.38. Dengan adanya fitur SMS ini maka perangkat Stasiun Seluler Digital (Digital Cellular Terminal, seperti ponsel) dapat mengirim dan menerima pesan-pesan teks dengan panjang sampai dengan 160 karakter melalui jaringan GSM.

Selama berada pada jangkauan pelayanan GSM, hanya dalam beberapa detik SMS dapat dikirimkan ke perangkat Stasiun Seluler Digital lainnya. Lebih dari sekedar pengiriman pesan biasa, layanan SMS memberikan garansi SMS akan sampai pada tujuan meskipun perangkat yang dituju sedang tidak aktif yang dapat disebabkan karena sedang dalam kondisi mati atau berada di luar jangkauan layanan GSM. Pesan yang belum terkirim akan disimpan sementara oleh jaringan, dan akan segera dikirimkan ke perangkat yang dituju setelah adanya tanda kehadiran dari perangkat di jaringan tersebut.

Dengan fakta bahwa layanan SMS (melalui jaringan GSM) hingga jangkauan nasional dan internasional dengan waktu keterlambatan yang sangat kecil, sehingga dimungkinkan layanan SMS cocok untuk dikembangkan dalam aplikasi-aplikasi seperti: pager, e-mail, dan notifikasi voice mail, serta layanan pesan banyak pemakai (multiple users). Namun pengembangan aplikasi tersebut masih bergantung pada tingkat layanan yang disediakan oleh operator jaringan.

2.2.1. PDU (Protocol Data Unit) SMS

Dalam proses pengiriman atau penerimaan pesan pendek (SMS), data yang dikirim maupun diterima oleh stasiun bergerak menggunakan salah satu dari 2 mode yang ada, yaitu: mode teks, atau mode PDU (Protocol Data Unit). Dalam mode PDU, pesan yang dikirim berupa informasi dalam bentuk data dengan beberapa kepala-kepala informasi. Hal ini dipermudah jika dalam pengiriman akan dilakukan kompresi data, atau akan dibentuk sistem penyandian data dari karakter dalam bentuk untaian bit-bit biner. Senarai PDU tidak hanya berisi pesan teks saja, tetapi terdapat beberapa meta-informasi yang lainnya, seperti nomor pengirim, nomor SMS Centre, waktu pengiriman, dan sebagainya. Semua informasi yang terdapat dalam PDU, dituliskan dalam bentuk pasangan pasangan bilangan heksadesimal yang disebut dengan pasangan oktet.

Data yang mengalir ke/dari SMS-Centre harus berbentuk PDU (Protocol Data Unit). PDU berisi bilangan-bilangan heksadesimal yang mencerminkan bahasa I/O. PDU terdiri atas beberapa Header. Header untuk kirim SMS ke SMS-Centre berbeda dengan SMS yang diterima dari SMS-Centre.

Maksud dari bilangan heksa desimal adalah bilangan yang terdiri atas 0, 1, 2 , 3, 4, 5 ,6 ,7 ,8 ,9 ,A ,B ,C ,D ,E ,F. Sebagai contoh, untuk angka desimal 1000, bilangan heksa desimalnya adalah 3E8.

Cara mengkonversikanya :

1000 : 16 = 62 sisa 8 = 8

62 : 16 = 3 sisa 14 = E

3 : 16 = 0 sisa 3 = 3

2.2.1.1. Delapan Header Untuk Kirim SMS

PDU untuk mengirim SMS terdiri atas delapan header, sebagai berikut:

1. Nomor SMS-Centre

   Header pertama ini terbagi atas tiga bagian subheader, yaitu:

   a. Jumlah Pasangan Hexsadesimal SMS-Centre dalam bilangan heksa.

   b. Nasional/Internasional Code

      - untuk Nasional, kode subheader-nya yaitu 81

      - untuk Internasional, kode subheader-nya yaitu 91

   c. No SMS-Centre-nya sendiri, dalam pasangan heksa dibalik-balik. Jika tertinggal satu angka heksa yang tidak memiliki pasangan, angka tersebut akan dipasangkan dengan huruf F didepannya.

      Contoh: untuk nomor SMS-Centre Excelcom dapat ditulis dengan dua cara sebagi berikut:

      Cara pertama:

      0818445009 diubah menjadi:

      a. 06 à karena 1 + 5 = 6 pasang

      b. 81 à 1 pasang

      c. 80-81-44-05-90 à 5 pasang

      Digabung menjadi: 06818081440590

      Cara kedua:

      0818445009 diubah menjadi:

      a. 07 à karena 1 + 6 = 7 pasang

      b. 91 à 1 pasang

c. 26-18-48-54-00-F9à 6 pasang

Digabung menjadi: 07912618485400F9

Berikut ini berupa beberapa no SMS-Center operator seluler di Indonesia.

Tabel 2-1. Tabel SMS Center Operator Di Indonesia

Cara pertama:

No	Operator Selular	SMS-Centre No	Kode PDU
1.	Telkomsel	0811000000	06818011000000
2.	Satelindo	0816124	0581806121F4
3	Excelcom	0818445009	06818081440590
4.	Indosat-M3	0855000000	06818055000000

Cara kedua:

No	Operator Selular	SMS-Centre No	Kode PDU
1.	Telkomsel	62811000000	07912618010000F0
2.	Satelindo	62816124	059126181642
3	Excelcom	62818445009	07912618485400F9
4.	Indosat-M3	62855000000	07912658050000F0

2. Tipe SMS

   Untuk tipe SEND tipe SMS=1. Jadi bilangan heksanya adalah 01.

3. Nomor Referensi SMS

   Nomor referensi ini dibiarkan dulu 0, jadi bilangan heksanya adalah 00. Nanti akan diberikan sebuah nomor referensi otomatis oleh handphone atau alat SMS-gateway.

4. Nomor Handphone Penerima

Sama seperti cara menulis PDU Header untuk SMS-Center, header ini juga terbagi atas tiga bagian, sebagi berikut:

1. Jumlah bilangan desimal nomor ponsel yang dituju dalam bilangan heksa
2. National/international Code.
   - ✓ Untuk nasional, kode subheader-nya: 81
   - ✓ Untuk internasional, kode subheader-nya: 91
3. Nomor handphone yang dituju, dalam pasangan heksa dibalik-balik. Jika tertinggal satu angka heksa yang tidak memiliki pasangan, angka tersebut dipasangkan dengan huruf F didepannya.

   Contoh:

   Untuk nomor handphone yang dituju = 628129573337 dapat ditulis dengan dua cara sebagi berikut:

   Cara pertama: 08129573337 diubah menjadi:

   a. 0B : ada 11 angka

   b. 81

   c. 80-21-59-37-33-F7

   Digabung menjadi: 0B818021593733F7

   Cara kedua: 628129573337 diubah menjadi:

   a. 0C : ada 11 angka

   b. 91

   c. 26-18-92-75-33-73

   Digabung menjadi: 0C91261892753373

5. Bentuk SMS, antara lain:

   0 -> 00 -> dikirim sebagi SMS

   1 -> 01 -> dikirim sebagai telex

   2 -> 02 -> dikirim sebagai fax

   Dalam hal ini, untuk mengirim dalam bentuk SMS tentu saja kita memakai 00.

6. Skema Encoding Data I/O

   Ada dua skema, yaitu:

   Skema 7 bit ditandai dengan angka 0 = 00

   Skema 8 bit ditandai dengan angka lebih besar dari 0 diubah ke heksa

   Kebanyakan handphone/SMS Gateway yang ada di pasaran sekarang menggunakan skema 7 bit sehingga kita menggunakan kode 00.

7. Jangka Waktu sebelum SMS Expired

   Jika bagian ini di-skip, itu berarti kita tidak membatasi waktu berlakunya SMS. Sedangkan jika kita isi dengan suatu bilangan integer yang kemudian diubah ke pasangan heksa tertentu, bilangan yang kita berikan tersebut akan mewakili jumlah waktu validitas SMS tersebut.

Tabel 2-2. Rumus Untuk Menghitung Jangka Waktu Validasi SMS

Integer (INT)	Jangka Waktu Validasi SMS
0-143	(INT +1) menit (berarti:5 menit s/d 12 jam)
144-167	12 jam + (INT-143) x 30 menit
168-196	(INT – 166) x 1 hari
197-255	(INT – 192) x 1 minggu

Agar SMS kita pasti terkirim sampai ke handphone penerima, sebaiknya kita tidak memberikan batasan waktu validnya.

8. Isi SMS

Header ini terdiri atas dua subheader, yaitu:

a. Panjang isi (jumlah huruf dari isi)

Misalnya untuk kata "hello" ada 5 huruf -> 05

b. Isi berupa pasangan bilangan heksa

Untuk handphone / SMS Gateway berskema encoding 7 bit, jika kita mengetikan suatu huruf dari keypad-nya, berarti kita telah mebuat 7 angka 1/O berurutan.

Skema 7 bit tersebut diperlihatkan pada dibawah ini :

Ada dua langkah yang harus kita lakukan untuk mengkonversi isi SMS, yaitu:

Langkah Pertama : mengubahnya menjadi kode 7 bit

Langkah kedua : mengubahnya kode 7 bit menjadi 8 bit, yang diwakili oleh pasangan heksa.

Contoh: untuk kata "hello"

Langkah pertama:

Bit 7 1

h 110 1000

e 110 0101

l 110 1100

l 110 1100

o 110 1111

Langkah kedua:

E 8

h 1110 1000

3 2

e 0011 0010 **1**

9 B

l 1001 1011 **00**

F D

l 1111 1101 **100**

0 6

o 0000 0110 **1111**

Oleh karena itu total 7 bit x 5 huruf=35 bit, sedangkan yang kita perlukan adalah 8 bit x 5 huruf=40 bit, maka diperlukan 5 bit dummy yang diisi dengan bilangan0.

Setiap 8 bit mewakili suatu pasangan heksa. Tiap 4 bit mewakili suatu angka heksa, tentu saja karena secara logika $2^4 = 16$.

Dengan demikian kata "hello" hasil konversinya menjadi: **E8329BFD06**

9. Menggabungkan kedelapan header

Setelah kita mempelajari masing-masing header maupun subheader untuk mengirim SMS di atas, kini kita akan menggabungkannya menjadi PDU yang lengkap.

Contoh:

Untuk mengirimkan kata "hello" ke nomor handphone 618129573337 lewat SMS-Centre Excelcom, tanpa membatasi jangka waktu valid, maka PDU lengkapnya adalah:

07912618485400F901000C91261892753373000005E8329BFD06

(07912618485400F9,01,00,0C91261892753373,00,00,05E8329BFD06)

Tabel 2-3. Tabel Skema 7 bit (ASCII)

				b7	0	0	0	0	1	1	1	1
				B6	0	0	1	1	0	0	1	1
				B5	0	1	0	1	0	1	0	1
b4	b3	b2	B1		0	1	2	3	4	5	6	7
0	0	0	0	0	@	D	SP	0	-	P		p
0	0	0	1	1			!	1	A	Q	a	q
0	0	1	0	2		F	"	2	B	R	b	r
0	0	1	1	3	$	G	#	3	C	S	c	s
0	1	0	0	4		L		4	D	T	d	t
0	1	0	1	5		W	%	5	E	U	e	u
0	1	1	0	6		P	&	6	F	V	f	v
0	1	1	1	7		Y	'	7	G	W	g	w
1	0	0	0	8		S	(	8	H	X	h	x
1	0	0	1	9		Q	)	9	I	Y	i	y
1	0	1	0	10	LF	X	*	:	J	Z	j	z
1	0	1	1	11			+	;	K	Ä	k	ä
1	1	0	0	12			,	<	L	Ö	l	ö
1	1	0	1	13	CR		-	=	M		m	
1	1	1	0	14		b	.	>	N	Ü	n	ü
1	1	1	1	15			/	?	O		o	

2.2.1.2. PDU untuk SMS Terima dari SMS-Centre

Delapan Header untuk SMS-Terima. Kebanyakan header dibawah ini telah dibahas sebelumnya, kecuali beberapa yang berbeda, dijelaskan di bawah ini:

1. No SMS-Centre.
2. Tipe SMS à untuk SMS-Terima = 4 -> **04**

3. Nomor handphone pengirim.
4. Bentuk SMS.
5. Skema encoding.
6. Tanggal dan waktu SMS di-stamo di SMS-Centre

   Diwakili oleh 12 bilangan heksa(6 pasang) yang berarti:yy/mm/dd hh:mm:ss

   Contoh: 207022512380à 01/07/22 15:32:08à 22 Juli 2002 15:32:08 Wib
7. Batas validasi waktu jika tidak dibatasi dilambangkan dengan **00**
8. Isi SMS.

Membedah Kedelapan Header

Setelah mengupas satu demi satu header untuk SMS-Terima ini, maka untuk PDU dibawah ini:

07912658050000F0040C912658161073980000207022512380000 5E8329BFD06

07912658050000F0,04,0C91265816107398,00,00,207022512380,00,05E8329BFD06

Dapat diartikan sebagai berikut:

1. SMS tersebut dikirim lewat SMS-Centre:62855000000
2. SMS tersebut merupakan SMS-Terima
3. SMS tersebut dikirim dari hadnphone dengan nomor *Sim Card*:628561013789
4. SMS tersebut diterima dalam bentuk SMS
5. SMS tersebut memiliki skema encoding 7 bit
6. SMS tersebut sampai di SMS-Centre pada tanggal 22-07-02, pukul: 15:32:08 Wib
7. SMS tersebut tidak memiliki batas waktu valid
8. SMS tersebut isinya adalah "hello".

2.3. Hp Siemens

2.3.1. Pinout HP siemens

Untuk melakukan koneksi dengan mikrokontroler, pada HP Siemens terdapat pin out dengan susunan seperti yang ditunjukan dalam tabel berikut dibawah ini :

Tabel 2-4. Pinout HP Siemens

Pin	Nama	Fungsi	In/Out
1	GND	Ground	
2	Self Service	Recognition / control battery	In
3	Load	Chargimg Voltage	Out
4	Battery	Battery	In
5	Data Out	Data Sent	Out
6	Data In	Data Received	Out
7	Z_Clk	Recognition / control accessories	In
8	Z_Data	Recognition / control accessories	
9	MICG	Ground for microphone	
10	MIC	Microphone input	In
11	AUD	Loudspeaker	
12	AUDG	Ground for eksternal LS	Out

2.3.2. Kabel Data Siemens Seri M35

Siemens seri M35 dilengkapi dengan kabel data untuk melakukan komunikasi data secara serial dengan terminal lain. Kecepatan transfer data sebesar19200. Untuk mengadakan komunikasi serial, pin-pin yang digunakan adalah sebagai berikut:

Tabel 2-5. Kabel Data Siemens M35

Nomor Pin	Nama	Fungsi
1	GND	Ground
5	TFMS/DFMS – Terminal adaptor equipment From Mobile Station Data From Mobile Station	Serial Data Out (Out)
6	TTMS/DTMS – Terminal adaptor equipment To Mobile Station Data To Mobile Station	Serial Data In (RX)

2.3.3. Perintah AT Command

Perintah AT (Hayes AT Command) digunakan untuk berkomunikasi dengan terminal (modem) melalui gerbang serial pada komputer. Dengan menggunakan perintah AT dapat diketahui atau dibaca kondisi dari terminal seperti mengetahui kondisi sinyal, kondisi baterai, mengirim pesan, membaca pesan, menambah item pada daftar telepon, dan sebagainya. Berikut dibawah ini adalah beberapa jenis perintah AT yang berhubungan dengan penanganan pesan-pesan SMS.

AT+CMGS, perintah ini digunakan untuk mengirim pesan

AT+CMGR, perintah ini digunakan untuk membaca pesan

AT+CMGF, perintah ini digunakan untuk format pesan

AT+CMGD, perintah ini digunakan untuk menghapus pesan

AT+CNMI, perintah ini digunakan untuk prosedur indikasi pesan baru

AT+CPMS, perintah ini digunakan untuk pemilihan target memori

AT+CSMS, perintah ini digunakan untuk pemilihan layanan pesan

2.4. Mikrokontroler AVR (Alf and Vegard's Risc Processors)

Atmel sebagai salah satu vendor yang mengembangkan dan memasarkan produk-produk mikroelektronika telah dijadikan sebagai suatu teknologi standar bagi para desainer masa kini. Dengan perkembangan terkini adalah generasi AVR (Alf and Vegard's Risc Processors).

Mikrokontroler AVR adalah mikrokontroler dengan arsitektur RISC 8 bit, dimana semua instruksi dikemas dalam kode 16-bit (16-bits word) dan sebagian besar instruksi dieksekusi dalam 1 siklus clock, berbeda dengan instruksi MCS51 yang dieksekusi dalam 12 siklus clock. Kedua jenis mikrokontroler tersebut dibedakan oleh perbedaan arsitektur yang dimiliki. AVR berteknologi RISC (Reduced Instruction Set Computing), sedangkan seri MCS51 berteknologi CISC (Complex Instruction Set Computing).

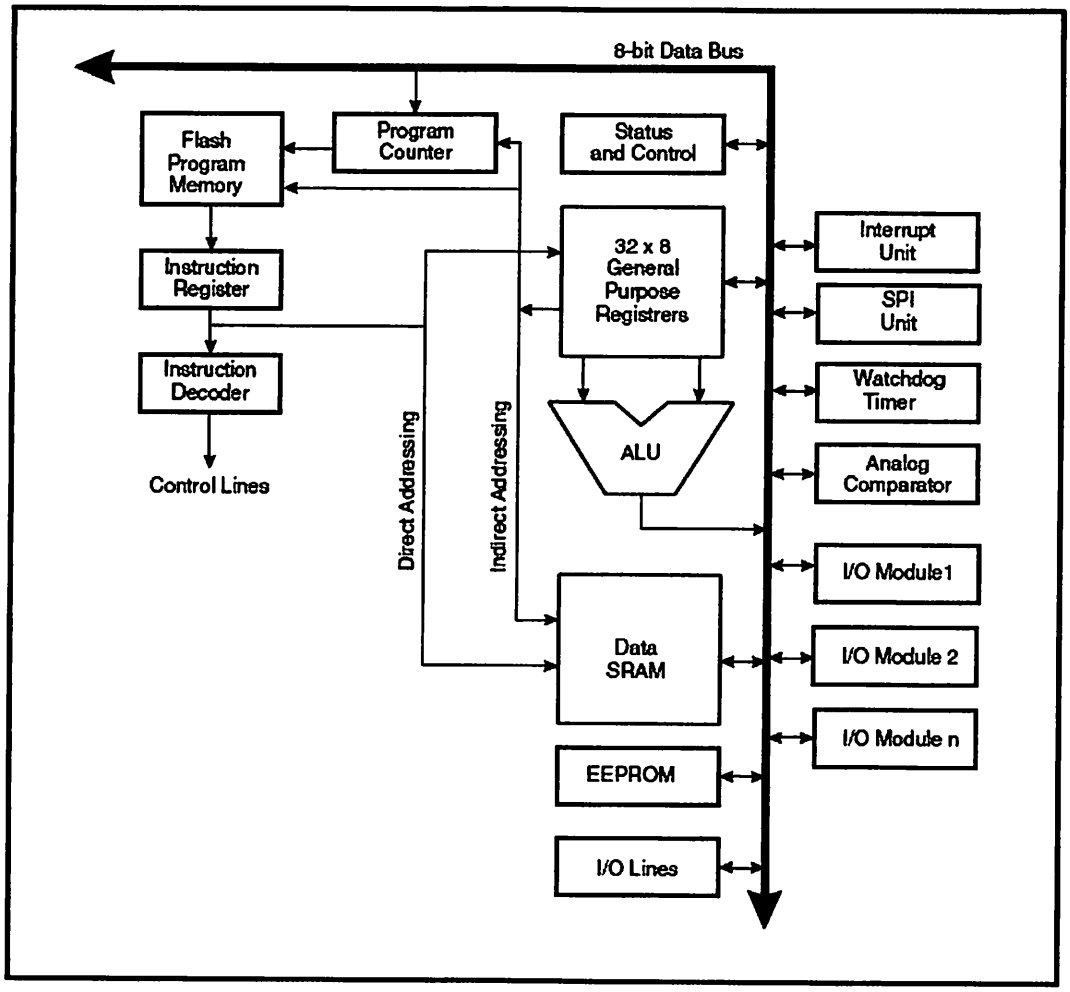
Secara umum, mikrokontroler jenis AVR dapat dikelompokkan menjadi 4 kelas, yaitu keluarga Atiny, keluarga AT90Sxx, keluarga ATMega, dan AT86RFxx. Pada dasarnya masing-masing kelas dibedakan oleh memori, peripheral, dan fungsinya. Dari segi arsitektur dan instruksi yang dipergunakan, mereka bisa dikatakan hampir sama. Berdasarkan uraian tersebut diatas maka dalam tugas akhir ini digunakan mikrokontroler ATMega8535.

2.4.1. Arsitektur ATMega8535

ATMega8535 adalah mikrokontroler 8-bit berteknologi CMOS yang berdaya rendah berbasis AVR dengan arsitektur RISC. Dengan keuntungan eksekusi instruksi rata-rata satu siklus clock merupakan pendekatan 1 MIPS

(million instruction Per Second) memberikan perancangan sistem dengan konsumsi daya optimal versus kecepatan proses.

Inti kombinasi AVR adalah kombinasi instruksi yang beraneka ragam dengan 32 register (32 general purpose working register). Ke-32 register tersebut langsung dihubungkan ke Arithmetic Logic Unit (ALU), melalui dua register independen, sehingga dapat diakses dalam satu instruksi yaitu satu siklus clock. Hasil dari arsitektur ini lebih efisien sepuluh kali dari mikrokontroler jenis CISC seperti AT89Cxxx atau AT89Sxxx.



Gambar 2.1. Blok Diagram Arsitektur ATMega8535

Mikrokontroler ATMega8535 memiliki model arsitektur *harvard*, dimana memori untuk data dan program terpisah, bus untuk program dan bus untuk data juga terpisah. Dalam arsitektur ATMega8535, seluruh GPR (General Purpose Register [32-register]) terhubung langsung ke ALU (Arithmetic Logic Unit)/procesor. Sehingga eksekusi instruksi lebih cepat. Dalam satu siklus clock terdapat dua register independen yang dapat diakses oleh satu instruksi. Teknik yang digunakan adalah fetch during execution atau memegang sambil mengerjakan. Hal ini berarti, dua operan dibaca dari satu register, dilakukan eksekusi operasi, dan hasilnya disimpan kembali dalam satu register, semuanya dilakukan hanya dalam satu siklus clock.

Dari 32 register terdapat enam buah register yang dapat digunakan untuk pengalamatan tidak langsung 16-bit sebagai register pointer (penunjuk). Register tersebut memiliki nama khusus, yaitu X, Y, Z. Masing-masing terdiri dari sepasang register, seperti: X (R27:R26), Y (R29:R28), dan Z (R31:R30). ALU mendukung operasi bit, fungsi aritmetika dan logika antara register dengan register atau antara register dengan nilai konstan, atau hanya operasi satu register.

Untuk kontrol aliran program disediakan intruksi lompatan bersyarat dan tak bersyarat, intruksi Call (panggil), dapat ditempatkan di seluruh ruangan program. Kebanyakan instruksi ATMega8535 mempunyai format 16-bit word. Setiap alamat memori program mengandung sebuah instruksi 16 atau 32 bit.

Selama interupsi dan pemanggilan subrutin, alamat program counter (PC) di simpan kedalam Stack. Stack akan efektif diletakkan di SRAM, dan konsekuensinya ukuran stack dibatasi oleh ukuran SRAM dan penggunaan

SRAM. Setiap pengguna dapat menginisialisasi SP dalam rutin RESET (sebelum subrutin atau interupsi dieksekusi). Stack pointer SP dapat ditulis dan di baca dalam ruangan I/O.

Kapabilitas detail dari ATMega8535 adalah sebagai berikut:

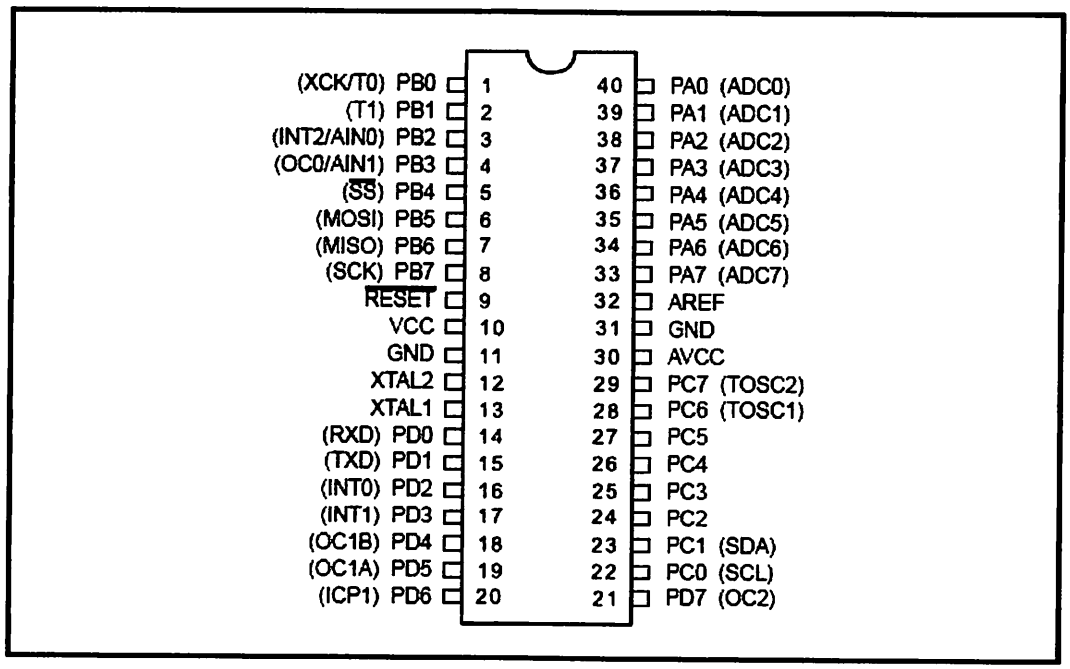
- Sistem mikroposesor 8 bit berbasis RISC dengan kecepatan maksimal 16 MHz.
- Kapabilitas memory flash 8 KByte, SRAM sebesar 512 Byte, dan EEPROM (Electrically Erasable Programmable Read Only Memory) sebesar 512 Byte.
- ADC internal dengan fidelitas 10 bit sebanyak 8 channel.
- Portal komunikasi serial (USART) dengan kecepatan maksimal 2,5 Mbps.
- Enam pilihan mode *sleep* dapat digunakan untuk menghemat penggunaan daya listrik.

2.4.2. Konfigurasi pin ATMega8535

Konfigurasi pin pada ATMega8535 adalah sebagai berikut:

1. VCC, merupakan pin yang digunakan sebagai pin masukan catu daya 5V.
2. GND, merupakan pin ground.
3. Port A (PA0..PA7), merupakan pin I/O dua arah atau dapat juga digunakan sebagai pin masukan ADC.
4. Port B (PB0..PB7), merupakan pin I/O dua arah dan dapat digunakan sebagai pin fungsi khusus, *Timer/Counter*, komparator analog, dan SPI.
5. Port C (PC0..PC7), merupakan pin I/O dua arah dan dapat digunakan sebagai pin fungsi khusus TWI, komparator analog dan *Timer* Oscilator.

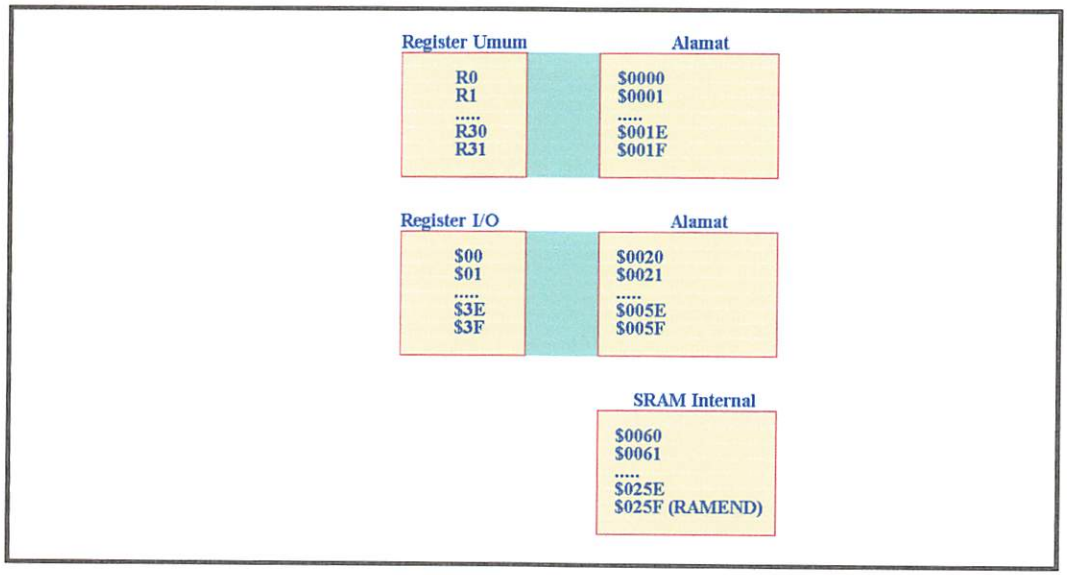
6. Port D (PD0..PD7), merupakan pin I/O dua arah dan dapat digunakan sebagai pin fungsi khusus komparator analog, interupsi eksternal, dan komunikasi serial.
7. RESET, merupakan pin yang digunakan untuk me-reset mikrokontroler.
8. XTAL1 dan XTAL2, merupakan pin yang digunakan sebagai masukan untuk clock eksternal.
9. AVCC, merupakan pin yang digunakan sebagai masukan tegangan ADC.
10. AREF, merupakan pin yang digunakan sebagai masukan tegangan referensi ADC.
11. Masukan dan keluarannya adalah kondisi high (tegangan 5V) dan kondisi Low (tegangan 0V).



Gambar 2.2. Konfigurasi Pin ATMega8535

2.4.3. Memori Data dan Memori Program ATMega8535

Ruang pengalamatan memori data dan memori program pada ATMega8535 di tempatkan secara terpisah. Memori data terbagi menjadi 3 bagian, yang terdiri dari 32 buah register umum, 64 buah register I/O, dan 512 *Byte* SRAM Internal. Register-register yang dapat digunakan untuk keperluan umum menempati s*pace* data pada alamat terbawah $00 sampai $1F.

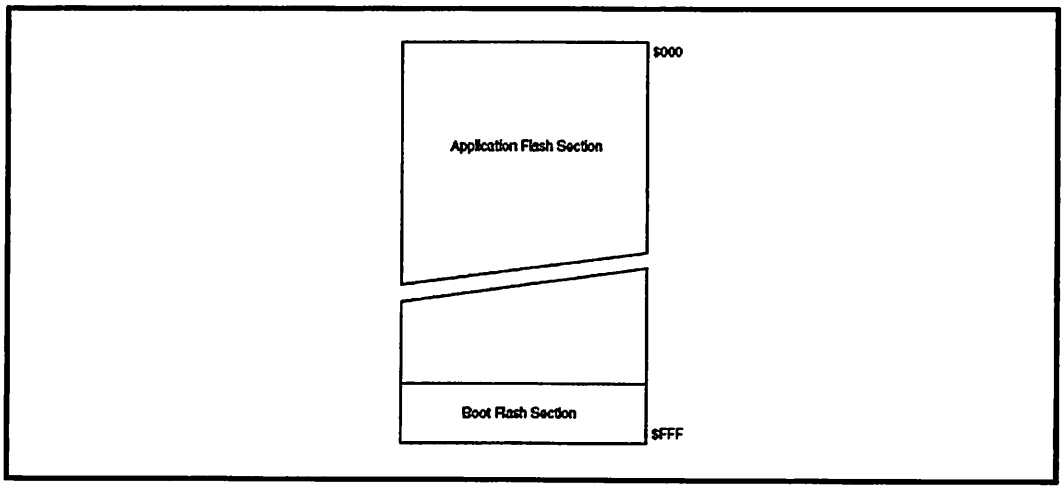


Gambar 2.3. Memori Data ATMega8535

64 alamat berikutnya yaitu mulai $20 hingga $5F adalah register-register khusus untuk menangani I/O dan kontrol terhadap mikrokontroler. Register tersebut merupakan register yang khusus digunakan untuk mengatur fungsi terhadap berbagai peripheral mikrokontroler, seperti kontrol register, timer/ counter, fungsi-fungsi I/O dan sebagainya. Alamat memori berikutnya pada lokasi $60 sampai $25F digunakan untuk SRAM 512 Byte

Memori program yang terletak dalam Flash PEROM disusun dalam word atau 2 Byte karena setiap instruksi lebarnya 16-bit atau 32-bit. Flash PEROM

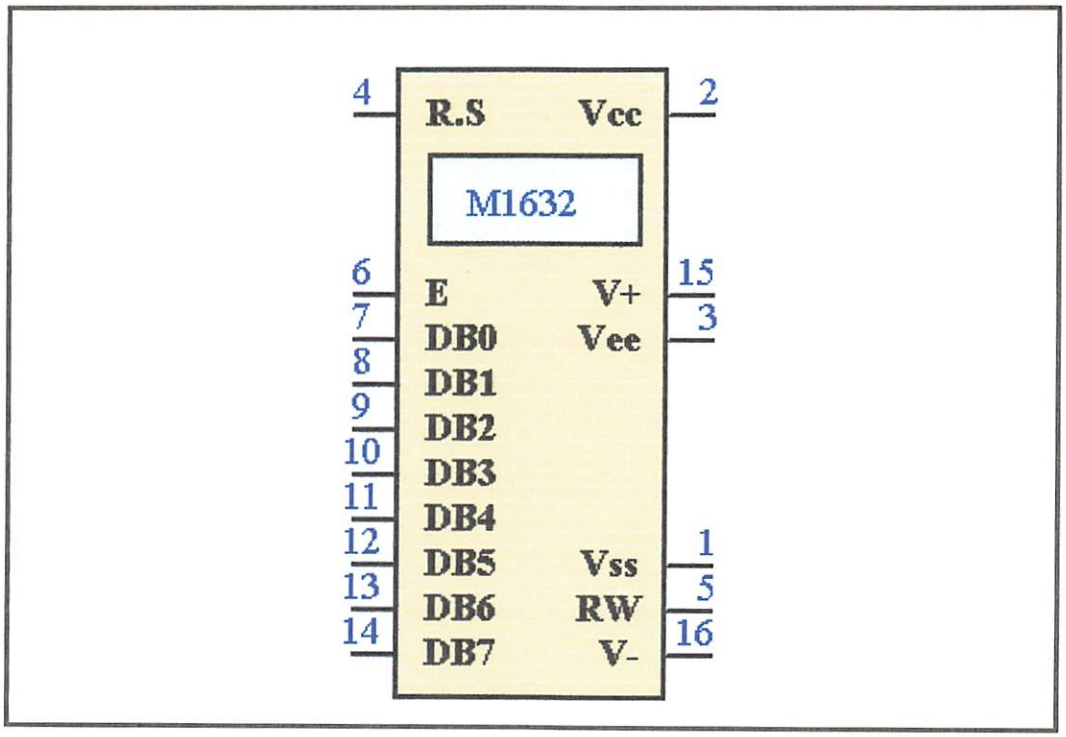
yang dimiliki AVR ATMega8535 adalah 4 KByteX16-bit dengan alamat mulai dari $000 sampai $FFF. AVR ATMega8535 dilengkapi dengan 12-bit Program Counter (PC) isi Flash dapat dialamati. Selain itu, AVR ATMega8535 juga memiliki memori data berupa EEPROM 8-bit sebanyak 512 Byte. Alamat EEPROM dimulai dari $000 sampai $1FF.



Gambar 2.4. Memori Program ATMega8535

2.5. LCD M1632

LCD Display Module M1632 buatan Seiko Instrument Inc. terdiri dari dua bagian, yang pertama merupakan panel LCD sebagai media penampil informasi dalam bentuk huruf/angka dua baris, masing-masing baris dapat menampung 16 huruf/ angka. Bagian kedua merupakan sebuah sistem yang dibentuk dengan mikrokontroler yang ditempelkan dibalik pada panel LCD, berfungsi mengatur tampilan informasi serta berfungsi mengatur komunikasi M1632 dengan mikrokontroler yang memakai tampilan LCD tersebut.



Gambar 2.5. LCD M1632

Konfigurasi Pin LCD M1632:

1. Vcc : +5V
2. GND: 0V
3. VEE : Tegangan Kontras LCD
4. RS : Register Select
5. R/W : 1=read, 0=write
6. E : Enable Clock LCD
7. D0 : Data Bus 0
8. D1 : Data Bus 1
9. D2 : Data Bus 2
10. D3 : Data Bus 3
11. D4 : Data Bus 4

12. D5 : Data Bus 5

13. D6 : Data Bus 6

14. D7 : Data Bus 7

15. Anode : Tegangan positif backlight

16. Katode: Tegangan negatif backlight

Untuk dapat berhubungan dengan mikrokontroler pemakai, LCD M1632 dilengkapi dengan 8 jalur data (DB0..DB7) yang digunakan untuk menyalurkan kode ASCII maupun perintah pengatur kerjanya M1632. Selain itu dilengkapi pula dengan E, R/W dan RS. Kombinasi lainya E dan R/W merupakan sinyal standar buatan Motorola. Proses mengirim/mengambil data ke/dari M1632 dijabarkan sebagai berikut:

1. RS harus dipersiapkan dulu, untuk menentukan jenis data yang dikirim ke M1632.
2. R/W di-nol-kan untuk menandakan akan diadakan pengiriman data ke M1632. Data yang akan dikirim disiapkan di DB0..DB7, sesaat kemudian sinyal E di-satu-kan dan di-nol-kan kembali. Sinyal E merupakan sinyal sinkronisasi, saat E berubah dari 1 menjadi 0 data di DB0 .. DB7 diterima oleh M1632.
3. Untuk mengambil data dari M1632 sinyal R/W di-satu-kan, menyusul sinyal E di-satu-kan. Pada saat E menjadi 1, M1632 akan meletakkan datanya di DB0 .. DB7, data ini harus diambil sebelum sinyal E di-nol-kan kembali.

Setelah diberi sumber daya, ada beberapa langkah persiapan yang harus dikerjakan dulu agar M1632 bisa digunakan.

Langkah-langkah tersebut antara lain adalah :

1. Tunggu dulu selama 15 mili-detik atau lebih.
2. Mengirim perintah 30h, artinya transfer data antara M1632 dan mikrokontroler dilakukan dengan mode 8 bit.
3. Tunggu selama 4.1 mili-detik.
4. Kirimkan sekali lagi perintah 30h.

   Tunggu lagi selama 100 mikro-detik.

2.6. Relay

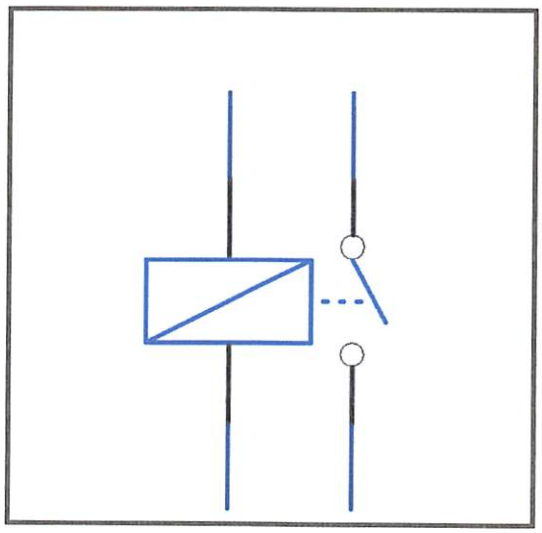
Relay merupakan salah satu jenis saklar magnetik yang dapat memutuskan atau menghubungkan kontak-kontak dari jarak jauh dengan arus. Sebuah relay terdiri dari satu kumparan dan inti yang mana bila dialiri arus kumparan tersebut akan menjadi magnet dan menutup atau membuka kontak-kontak. Keuntungan dari relay adalah dapat menghubungkan daya yang besar dengan memberi daya yang kecil pada kumparannya. Relay digolongkan berdasarkan arusnya menjadi dua yaitu:

a. Relay arus searah (DC Relay)
b. Relay arus bolak-balik (AC Relay)

Dalam pemilihan suatu relay harus diketahui sifat dari relay tersebut, yaitu :

- Arus kerja apakah arus AC atau DC
- Hambatannya tergantung dari banyaknya lilitan dan penampang lilitan, dimana nilainya mulai 1Ω – 50.000Ω.
- Arus tarik, agar jangkar dapat tertarik harus diperhitungkan arus tarik kumparannya.

- Tegangan tarik, tegangan yang diperlukan sepanjang kumparan agar dapat menarik jangkar (hasil kali arus searah dan hambatan).
- Jenis dan jumlah kontak, yaitu kontak penghubung atau kontak pemutus.
- Kemampuan hantar arus kontak, yaitu batas kemampuan suatu kontak untuk menghantarkan suatu arus secara terus-menerus tanpa menimbulkan kerusakan.
- Tegangan maksimal, yaitu tegangan terbesar yang mampu dikenakan pada kontaknya dipengaruhi oleh jarak kontaknya.



Gambar 2.10. Simbol relay

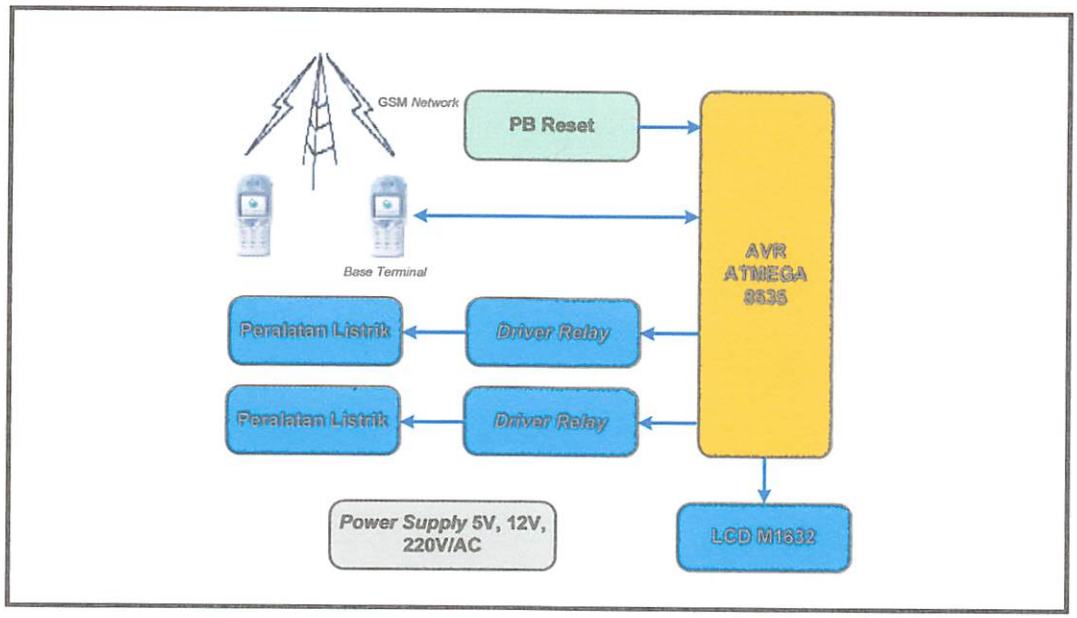
BAB III
PERENCANAAN SISTEM

Pada bab ini akan dibahas perancangan alat Programable Timer peralatan listrik diakses via SMS beserta prinsip kerja keseluruhan sistem. Perancangan alat dalam sistem ini terdiri perancangan perangkat keras (hardware) dan perangkat lunak (software).

Tujuan dari Tugas Akhir ini adalah untuk merancang alat programable timer peralatan listrik diakses via SMS. Sistem ini dibuat untuk mengendalikan keaktifan peralatan listrik menggunakan timer yang diakses via SMS dengan mikrokontroller sebagai pengendali.

3.1. Blok Diagram Sistem

Blok diagram programable timer peralatan listrik diakses via SMS yang direncanakan ditunjukkan pada gambar 3.1.



Gambar 3.1. Blok Diagram Sistem

3.2. Prinsip Kerja Alat

Programmable Timer Peralatan Listrik diakses Via SMS ini dirancang untuk memiliki proses kerja sebagai berikut:

1. User mengetikkan perintah-perintah teks pada User Terminal.
2. Pesan perintah dalam bentuk SMS dikirim dari terminal pengguna (User Terminal) menuju ke terminal utama (Base Terminal) melewati jaringan GSM.
3. Pesan yang diterima oleh terminal utama (Base Terminal) yang berupa PDU SMS langsung diberitahukan ke sistem mikrokontroller dan dibaca.
4. Mikrokontroller menterjemahkan pesan pada PDU SMS.
5. Mikrokontroller membaca perintah-perintah teks yang ada, kemudian menterjemahkan perintah tersebut menjadi tindakan pengendalian seting timer peralatan listrik.

3.3. Perencanaan Perangkat Keras (Hardware)

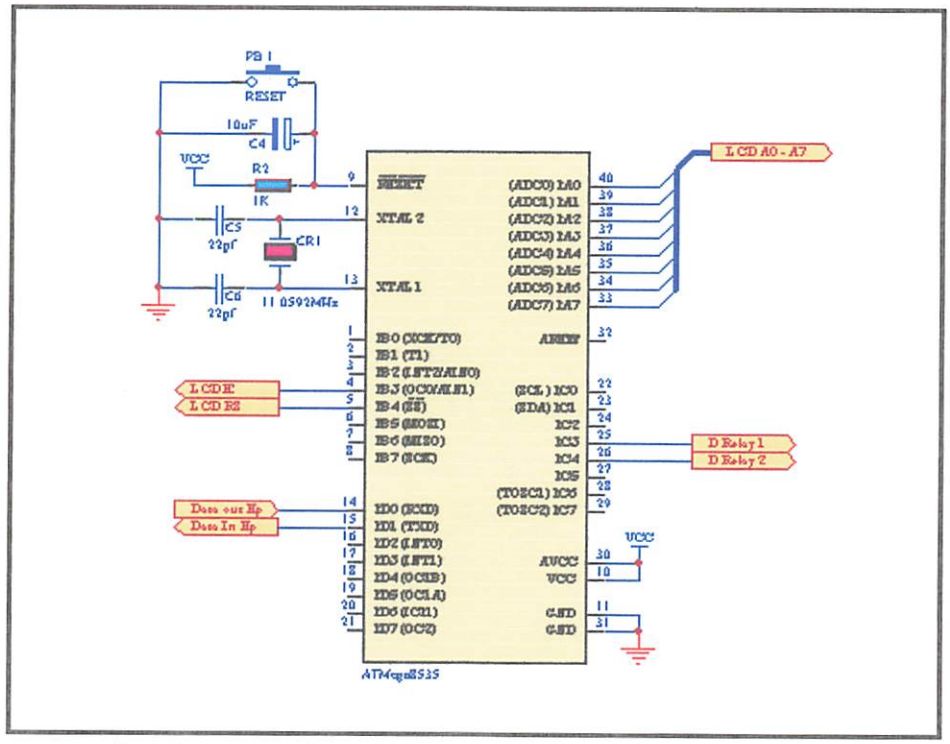
Perangkat keras yang digunakan dalam sistem ini terdiri dari beberapa bagian yaitu:

1. Sistem mikrokontroller AVR ATMega8535.
2. Handphone Siemens M35 beserta kabel datanya.
3. Rangkaian Driver Relay sebagai penghubung peralatan listrik yang di kontrol dengan tegangan jala-jala dari PLN AC220V.
4. Rangkaian LCD M1632 sebagai display timer dan kondisi peralatan listrik ON/OFF.
5. Rangkaian Power Supply dibutuhkan sebagai sumber tegangan kerja.

3.3.1 Sistem Mikrokontroller AVR ATMega8535

Mikrokontroller tipe ATMega8535 adalah produksi Atmel yang berbasis pada arsitektur AVR. Yang memiliki arsitektur RISC 8 bit, dimana semua instruksi dikemas dalam kode 16-bit (16-bits word) dan sebagian besar instruksi dieksekusi dalam 1 siklus clock serta dilengkapi dengan EEPROM (Electrically Erasable Programmable Read Only Memory) sebesar 512 byte.

Dalam rangkaian sistem ini terdapat komponen yang paling penting yaitu rangkaian mikrokontroller AVR ATMega8535. Sistem mikrokonroller berfungsi sebagai pengendali utama dan pemroses data antara HP, LCD M1632 dan driver relay. Perencanaan mikrokontroller dapat dilihat pada gambar 3.2.



Gambar 3.2. Sistem Mikrokontroller AVR ATMega8535

Agar sebuah mikrokontroller dapat bekerja sebagai pengontrol, maka kaki-kaki/port mikrokontroller dihubungkan dalam rangkaian-rangkaian eksternal. Dalam perancangan ini, port yang digunakan adalah sebagai berikut:

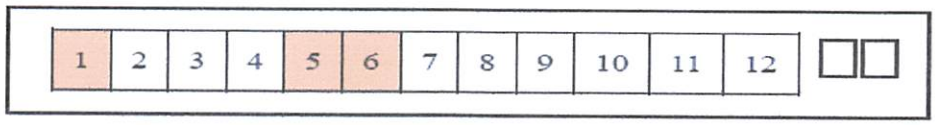
1. Port A dikonfigurasikan sebagai outputan:
   a. Port A0-A7 digunakan sebagai jalur data (bus) LCD.
2. Port B digunakan sebagai jalur Port I/O dimana pin yang digunakan adalah:
   a. PB4 dihubungkan dengan Pin E pada LCD sebagai enable clock LCD, logika 1 setiap kali pengiriman data..
   b. PB3 dihubungkan dengan Pin RS pada LCD untuk mengatur register select LCD dimana 0 = register perintah, sedangkan 1 = register data..
3. Port C merupakan Port I/O dimana pin yang digunakan adalah:
   a. PC3 dihubungkan dengan rangkaian driver relay 1 untuk mengatur keaktifan peralatan listrik 1 .
   b. PC4 dihubungkan dengan rangkaian driver relay 2 untuk mengatur keaktrifan peralatan listrik 2.
4. Port D merupakan Port I/O dimana pin yang digunakan adalah:
   a. PD0 (RXD) digunakan sebagai serial input yang dihubungkan dengan pin transmitter ponsel.
   b. PD1 (TXD) digunakan sebagai serial output yang dihubungkan dengan pin reciever ponsel.
5. XTAL1 dan XTAL2 digunakan sebagai input dari rangkaian osilator kristal. Rangkaian osilator kristal terdiri dari kristal osilator 11,0592 MHz,

kapasitor C1 dan C2 yang masing-masing bernilai 22 pF, akan membangkitkan pulsa clock yang menjadi penggerak bagi seluruh operasi internal mikrokontoller.

6. VCC dihubungkan dengan tegangan sebesar +5V yang berasal dari power supply yang sesuai dengan tegangan operasi chip tunggal yang diijinkan dalam data sheet.
7. GND dihubungkan ke ground catu daya.
8. Reset digunakan untuk mereset program kontrol mikrokontroller.

3.4. Antarmuka Komunikasi Serial

Koneksi dengan handphone Siemens pada dasarnya digunakan untuk dapat berkomunikasi dengan handPhone, sehingga diperlukan sebuah kabel data sebagai konektor untuk menghubungkan handphone dengan mikrokontroler. Konektor handphone Siemens M35i dapat dilihat pada gambar 3.3. Tidak semua pin out terhubung ke mikrokontroler, tetapi hanya pin nomor 1 (ground), 5 (Tx/data out) dan 6 (Rx/data in).



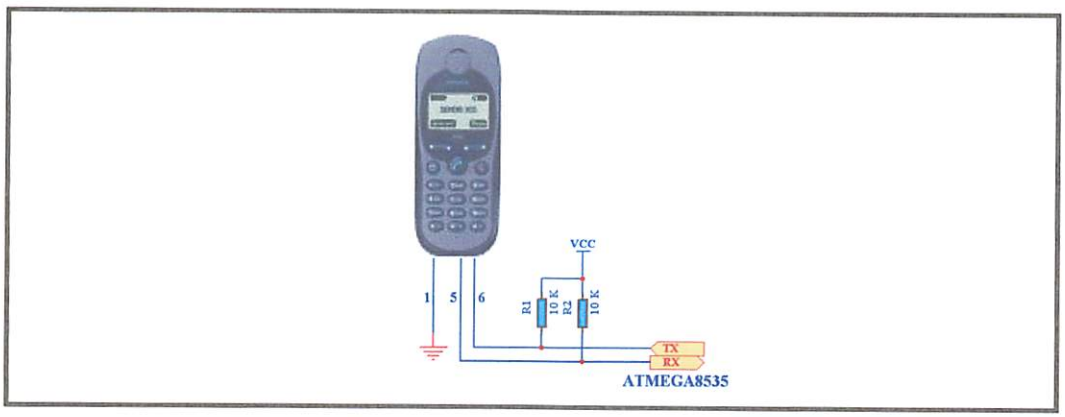
Gambar 3.3. Konektor Mobilephone M35i

Untuk penggunaan dari masing-masing pin dari konektor pada gambar dapat dilihat pada tabel berikut.

Tabel 3-1 Pin out Konektor M35i

PIN	NAMA	FUNGSI	IN/OUT
1	GND	Ground	
2	SELF SERVICE	Recognition/control battery charger	In/Out
3	LOAD	Charging Voltage	In
4	BATTERY	Battery	Out
5	DATA OUT	Data sent	Out
6	DATA IN	Data received	In
7	Z_CLK	Recognition/control accessories	
8	Z_DATA	Recognition/control accessories	
9	MICG	Ground for microphone	In
10	MIC	Microphone input	
11	AUD	Loudspeaker	Out
12	AUDG	Ground for eksternal speaker	

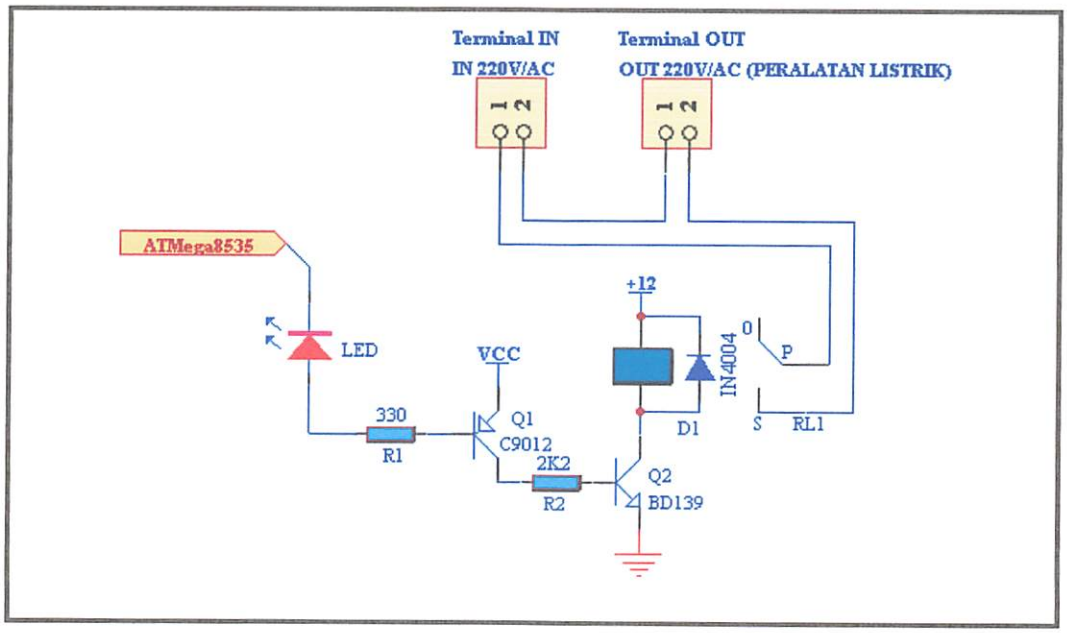
Pada perancangan sistem ini rangkaian konektor telepon seluler dengan sistem pengontrol utama (rangkaian mikrokontroller) ditunjukkan dalam gambar 3.4 pada perancangan ini, pin 5 yang berfungsi sebagai data out dihubungkan dengan PD0 (RXD), pin 6 yang berfungsi sebagai data recieved/data in dihubungkan pada PD1 (TXD) mikrokontroller, sedangkan pin 1 dihubungkan ke ground.



Gambar 3.4. Antarmuka Komunikasi Serial Mikrokontroller dengan HP

3.5. Rangkaian Driver Relay

Relay berfungsi untuk menghubungkan atau memutus aliran arus listrik yang dikontrol dengan memberikan tegangan dan arus tertentu pada koilnya. Pada perancangan ini digunakan relay DC dengan tegangan koil 12V DC, arus yang diperlukan sekitar 20-30mA. Karena itu kita tidak bisa langsung menghubungkan output dari mikrokontroler dengan relay karena arusnya tidak cukup besar sehingga perlu digunakan driver untuk penguat arus. Rangkaian driver pada alat ini dilengkapi dengan 'supression diode'. Diode ini berfungsi untuk mencegah 'kickback' yaitu transient yang terjadi pada koil relay (beban induktif) saat relay dimatikan. Gambar rangkaian driver relay ditunjukkan pada gambar 3.5.



Gambar 3.5. Rangkaian Driver Relay

Berikut adalah perhitungan perancangan pada rangkaian driver relay.

Dari data sheet relay diketahui :

$$R_{relay} = 400\Omega$$

$V_{relay} = 12$

Q1 adalah transistor PNP C9012 dengan h_{FE} 125

Q2 adalah transistor NPN BD139 dengan h_{FE} 100

Perhitungan pada Q2:

$$Ic_{sat2} = \frac{V_{relay}}{R_{relay}}$$

$$= \frac{12}{400}$$

$$= 30mA$$

$$Ib_{sat2} = \frac{Ic_{sat}}{h_{FE}}$$

$$= \frac{30mA}{100}$$

$$= 0{,}3 \text{ mA}$$

$$Rb_2 = \frac{Vb - Vbe}{Ib}$$

$$= \frac{5V - 0{,}7V}{0{,}3mA}$$

$$= 1{,}4K\Omega$$

Agar relay dapat bekerja dengan baik, perbedaan antara Ib dan Ib_{sar} harus lebih besar dari 1 ($Ib\text{-}Ib_{sat}>0{,}1$), Oleh karena itu pada perancangan ini besar Rb pada Q2 yang digunakan adalah sebesar 2K2. Sehingga besar arus Ib menjadi:

$$Ib_2 = \frac{Vb - Vbe}{R4}$$

$$= \frac{5 - 0{,}7}{2K2}$$

$= 1{,}954$ mA

Perhitungan pada Q1:

$Ic_{sat1} = Ib_{Q2}$

$= 1{,}954$ mA

$Ib_{sat1} = \frac{Ic_{sat}}{h_{FE}}$

$= \frac{1{,}954}{125}$

$= 0{,}015$ mA

$Rb_1 = \frac{Vb - Vbe}{Ib}$

$= \frac{5 - 0{,}7}{0{,}015}$

$= 286\Omega$

Agar transistor Q1 dapat bekerja dengan baik dalam mensaklar transistor Q2, maka perbedaan antara Ib dan Ib_{sar} harus lebih besar dari 1 ($Ib\text{-}Ib_{sat}>0{,}1$), dan selain itu arus Ib pada Q1 supaya dapat digunakan untuk menyalakan LED sebagai lampu indikator bahwa rangkaian driver yang dirancang dapat bekerja dengan baik, maka pada perancangan ini besar Rb pada Q1 yang digunakan adalah sebesar 330Ω.

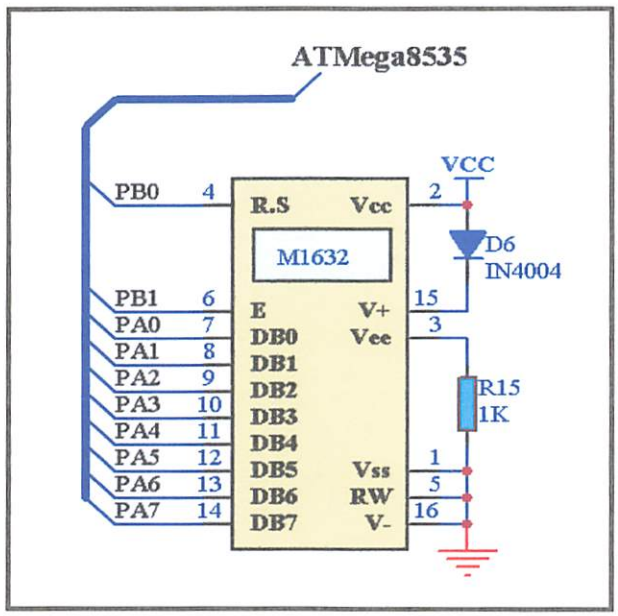
Pin PB,3 dan pin PD,6 pada mikrokontroller dihubungkan dengan driver relay sebagai masukan untuk mengaktifkan dan mematikan driver relay yang terhubung dengan peralatan listrik 1 dan peralatan listrik 2.

Driver relay berfungsi sebagai saklar ON/OFF pada peralatan listrik. Adapun prosesnya adalah:

1. Pada saat output mikrokontroler 'high', maka transistor PNP Q1 (C9012) dalam keadaan cut-off, sehingga arus Vcc tidak dapat mengalir menuju basis transistor NPN Q2 (BD139) dan LED juga tidak mendapatkan arus sehingga LED dalam keadaan OFF (mati), transistor Q2 dalam keadaan OFF (cut-off), relay OFF dan peralatan listrik juga dalam keadaan OFF.
2. Pada saat output mikrokontroler 'low', maka transistor PNP Q1 (C9012) dalam keadaan saturasi (ON), sehingga arus Vcc dapat mengalir menuju basis transistor NPN Q2 (BD139) dan LED juga akan mendapatkan arus sehingga LED dalam keadaan ON (nyala), transistor Q2 dalam keadaan ON (saturasi), relay ON dan peralatan listrik juga ON.

3.6. Rangkaian LCD M1632

Untuk dapat berhubungan dengan mikrokontroler LCD M1632 dilengkapi dengan 8 jalur data (DB0..DB7) yang digunakan untuk menyalurkan kode ASCII maupun perintah pengatur kerjanya M1632. Selain itu dilengkapi pula dengan E, R/W dan RS. Kombinasi lainya E dan R/W merupakan sinyal standar buatan Motorola. Gambar rangkaian LCD dapat dilihat pada gambar 3.6.



Gambar 3.6. Rangkaian LCD M1632

Berikut adalah tabel penjelasan fungsi pin-pin pada LCD M1632.

Tabel 3-2. Fungsi pin-pin pada LCD M1632

Nama Pin	Fungsi
DB0 – DB7	Merupakan saluran data, berisi perintah dan data yang akan ditampilkan pada LCD
E (Enable)	Sinyal operasi awal. Sinyal ini akan mengaktifkan data tulis atau baca
R/W	Sinyal seleksi tulis dan baca: 0 = tulis 1 = baca
RS	Sinyal pemilih register internal: 0 = instruksi register (tulis) / masukan data 1 = data register (tulis dan baca) / masukan instruksi
VEE	Untuk mengendalikan kecerahan LCD dengan mengubah-ubah nilai resistor variable yang dihubungkan padanya.
VCC	Catu daya +5V
VSS	Terminal ground

Proses mengirim/mengambil data ke/dari M1632 dijabarkan sebagai berikut :

1. RS harus dipersiapkan dulu, untuk menentukan jenis data yang dikirim ke M1632.

2. R/W di-nol-kan untuk menandakan akan diadakan pengiriman data ke M1632. Data yang akan dikirim disiapkan di DB0..DB7, sesaat kemudian sinyal E di-satu-kan dan di-nol-kan kembali. Sinyal E merupakan sinyal sinkronisasi, saat E berubah dari 1 menjadi 0 data di DB0 .. DB7 diterima oleh M1632.
3. Untuk mengambil data dari M1632 sinyal R/W di-satu-kan, menyusul sinyal E di-satu-kan. Pada saat E menjadi 1, M1632 akan meletakkan datanya di DB0 .. DB7, data ini harus diambil sebelum sinyal E di-nol-kan kembali.

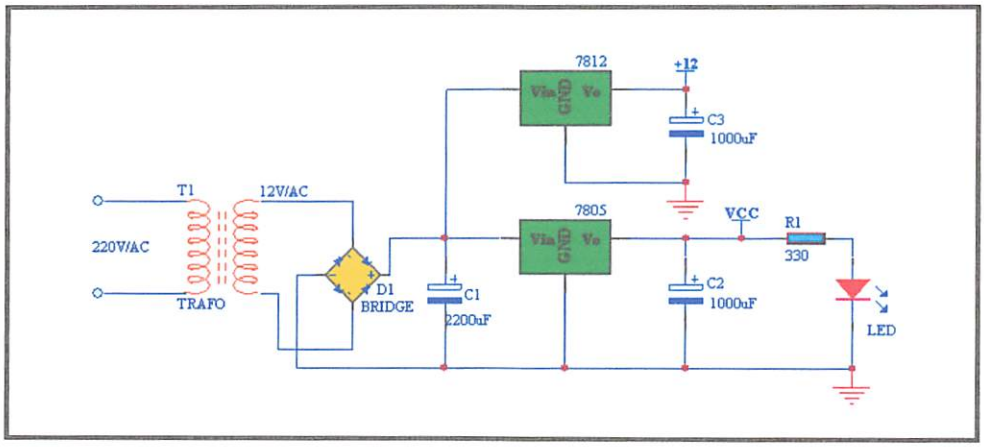
Pada perancangan LCD M1632 pin-pin yang digunakan adalah:

1. Pin DB0-DB7 digunakan sebagai 2 jalur input data yaitu digunakan sebagai input (menerima data) yang telah diproses dari mikrokontroller untuk ditampilkan pada LCD.
2. Pin RS pada LCD M1632 sebagai penentu jenis data yang akan dikirim, dihubungkan dengan PB1 mikrokontroller.
3. Pin E dihubungkan dengan PB0 mikrokontroller sebagai sinkronisasi LCD dengan mikrokontroller sebelum dan sesudah pengiriman data.

3.7. Rangkaian Power Supply

Rangkaian power supply dibutuhkan sebagai sumber tegangan kerja untuk keseluruhan rangkaian. Rangkaian power supply mendapatkan sumber tegangan dari tegangan jala-jala PLN sebesar 220V/AC. Tegangan 220V/AC ini kemudian diturunkan menjadi 12V/AC menggunakan transformator penurun tegangan. Tegangan AC 12V kemudian disearahkan oleh dioda bridge menjadi tegangan DC. Keluaran dari dioda bridge ini kemudian diinputkan ke IC regulator LM7805 dan LM7812 yang akan menghasilkan tegangan DC sebesar +5V dan +12V untuk

memberikan supply tegangan pada tiap-tiap rangkaian. Elco 2200uF dan 1000uF digunakan untuk membuang ripple pada tegangan DC. Led digunakan sebagai lampu indikator power supply. Gambar rangkaian power supply ditunjukkan pada gambar 3.7.



Gambar 3.7. Rangkaian Power Supply

3.8. Perencanaan Perangkat Lunak (software)

Perangkat lunak dirancang dengan menggunakan bahasa assembly yang ditulis dengan text editor AVR Studio versi 4. Dengan menggunakan software AVR Studio tersebut pula program sumber assembly di compile menjadi program objek maupun hexa, kemudian di load ke dalam mikrokontroler menggunakan software pony prog2000. Untuk memberikan gambaran umum jalannya program dan memudahkan pembuatan perangkat lunak, maka dibuat diagram alir yang dapat dilihat pada gambar 3.8.

Setiap SMS yang datang akan disimpan di SIM card pada lokasi memori mailbox dengan mengisi nomor mailbox yang paling rendah terlebih dahulu. Apabila seluruh lokasi mailbox sudah terisi, maka SMS center tidak akan mengirimkan SMS ke ponsel sampai tersedianya tempat kosong pada mailbox.

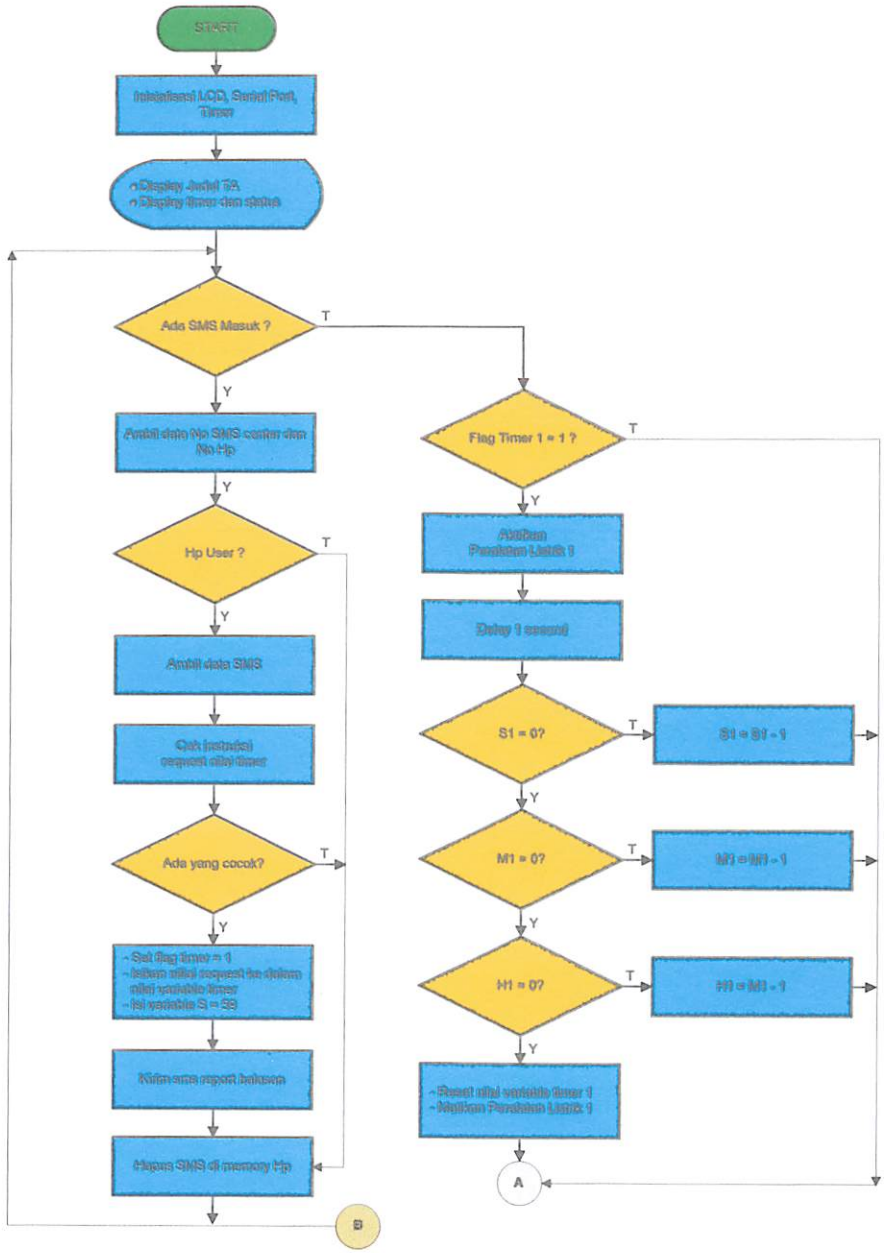
Pada rutin pembacaan SMS ini lokasi mailbox nomor 1 disiapkan untuk menampung SMS yang datang, dimana SMS tersebut dianggap sebagai perintah untuk seting timer peralatan listrik yang dikontrol. Setelah pembacaan SMS oleh mikrokontroler selesai, lokasi tersebut dihapus untuk menampung SMS yang lain.

Ponsel menerima SMS tidak sekedar berupa pesan, tetapi data pesan SMS tersebut merupakan bagian dari serangkaian data-data yang terbentuk saat pengiriman melalui SMS center. Apabila rangkaian data seluruhnya ditampung oleh mikrokontroler, maka dibutuhkan jumlah lokasi memori RAM yang besar. Hal itu tidak mungkin dilakukan karena terbatasnya besar RAM yang dimiliki oleh Mikrokontroler Atmega8535, yaitu sebesar 512 byte. Dengan demikian, maka data yang diterima akan langsung digunakan sebagai perintah untuk seting timer dan setelah itu langsung dihapus sehingga data tidak menumpuk.

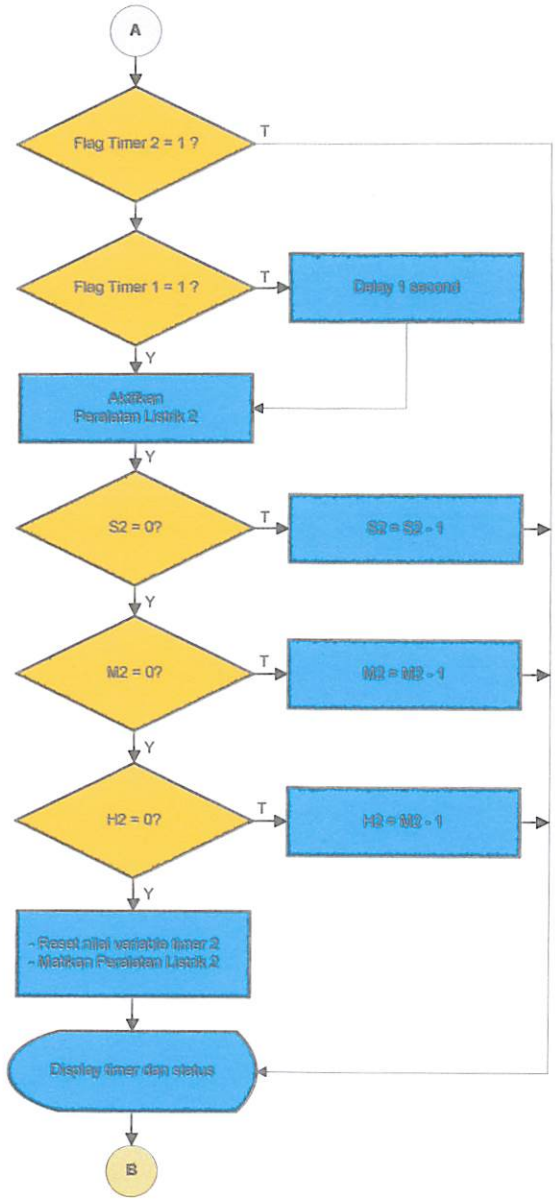
Setelah proses seting timer tersebut selesai, maka mikrokontroler akan secara otomatis mengaktifkan rangkaian driver relay sehingga peralatan listrik menjadi ON dan pada display LCD menampilkan report bahwa timer dalam keadaan ON dan peralatan listrik ON. Looping program mikrokontroler akan selalu melakukan pengecekan terhadap nilai timer yang di inputkan apakah telah habis atau belum. Apabila timer telah habis, maka mikrokontoler akan segera mematikan rangkaian driver relay sehingga peralatan listrik menjadi OFF dan pada display LCD akan menampilkan report bahwa timer telah habis dan peralatan listrik OFF. Kemudian kembali ke proses awal.

Alur proses pengendalian peralatan listrik ditunjukkan pada diagram alir gambar 3.8 dan 3.9.

Flowchart



Gambar 3.8. Flowchart Programmable Timer Peralatan Listrik diakses Via SMS



Gambar 3.9. Sambungan Flowchart Programmable Timer Peralatan Listrik diakses Via SMS

Keterangan :

S : Variable untuk menyimpan nilai detik

M : Variable untuk menyimpan nilai menit

H : Variable untuk menyimpan nilai jam

BAB IV
PENGUJIAN SISTEM

Pada bab ini akan dibahas mengenai pengujian sistem programmable timer peralatan listrik jarak jauh via SMS. Pengujian ini bertujuan untuk menguji apakah alat yang dibuat telah dapat bekerja sesuai perancangan. Pengujian dilakukan pada perangkat keras maupun perangkat lunak. Metode pengujian alat adalah sebagai berikut:

1. Pengujian Rangkaian Power Supply
2. Pengujian Rangkaian ATMega8535
3. Pengujian Rangkaian LCD
4. Pengujian Rangkaian RS232
5. Pengujian Rangkaian Driver Relay
6. Pengujian rangkaian secara keseluruhan

4.1. Pengujian Rangkaian Power Supply

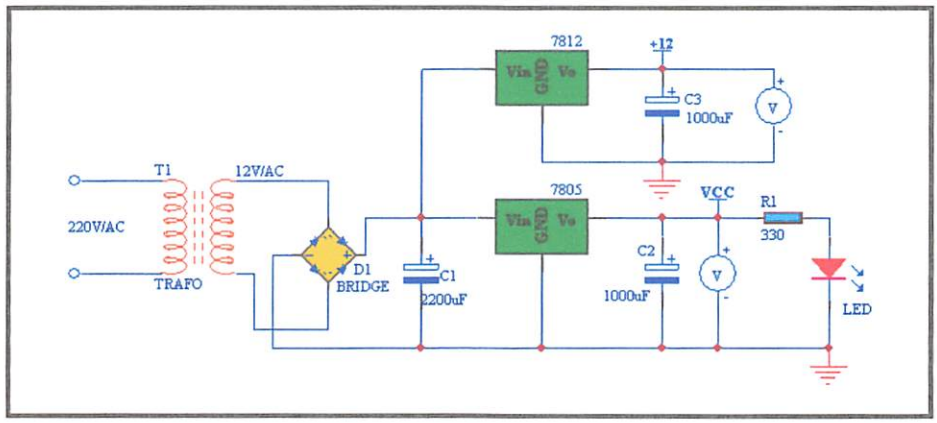
A. Tujuan

Untuk mengetahui tegangan yang dihasilkan oleh rangkaian power supply yang telah dirangkai. Dengan begitu dapat diketahui apakah terjadi kesalahan terhadap rangkaian power supply atau tidak.

B. Peralatan yang digunakan

1. 2 buah multimeter digital tipe DT-830B
2. Power supply.

C. Rangkaian pengujian



Gambar 4.1. Rangkaian Pengujian Tegangan Power Supply

Langkah – langkah pengujian

1. Merangkai modul rangkaian seperti pada gambar 4.1
2. Menseting Multimeter digital pada DC Volt dengan batas maksimal pengukuran yaitu 20 Volt DC.
3. Menghubungkan tegangan masukan pada rangkaian power supply dengan tegangan jala-jala dari PLN sebesar 220V/AC.
4. Mengaktifkan power supply.
5. Mengamati tegangan keluaran multimeter digital

D. Data hasil pengujian

Tabel 4-1. Hasil Pengukuran Rangkaian Power Supply

Indeks Pengujian	Out 7805 (V/DC)	Out 7812 (V/DC)
1	4,9	11,9
2	5,0	12,0
3	4,9	11,9
4	4,9	11,9

5	4,9	11,9
6	5,0	12,0
7	4,9	11,9
8	4,8	11,8
9	4,9	11,9
10	4,9	11,9

E. Analisa hasil

Terjadi perbedaan nilai dikarenakan oleh adanya beberapa faktor yang mempengaruhi, diantaranya kualitas dari tiap-tiap komponen yang digunakan nilainya tidak murni. Selain itu, tegangan jala-jala listrik yang digunakan tidak stabil. Berikut adalah perhitungan kesalahan nilai dari hasil pengukuran yang telah dilakukan:

Out 7805 :

$$\overline{X} = \frac{x_1 + x_2 + ... x_{10}}{n}$$

$$\overline{X} = \frac{4,9+5+4,9+4,9+4,9+5+4,9+4,8+4,9+4,9}{10}$$

$$\overline{X} = \frac{49,1}{10}$$

$$\overline{X} = 4,91$$

$$\%error = 100\% - \frac{\overline{X}}{5V} x100\%$$

$$\%error = 100\% - \frac{4,91}{5V} x100\%$$

$$\%error = 100\% - 98,2\%$$

$$\%error = 1,8\%$$

Out 7812 :

$$\overline{X} = \frac{x_1 + x_2 + ...x_{10}}{n}$$

$$\overline{X} = \frac{11{,}9+12+11{,}9+11{,}9+11{,}9+12+11{,}9+11{,}8+11{,}9+11{,}9}{10}$$

$$\overline{X} = \frac{119{,}1}{10}$$

$$\overline{X} = 11{,}91$$

$$\%error = 100\% - \frac{\overline{X}}{12V} x100\%$$

$$\%error = 100\% - \frac{11{,}91}{12V} x100\%$$

$$\%error = 100\% - 99{,}25\%$$

$$\%error = 0{,}75\%$$

F. Kesimpulan

Power supply dapat memberikan tegangan kerja untuk rangkaian keseluruhan sesuai dengan tegangan kerja yang dibutuhkan.

4.2. Pengujian Rangkaian ATMega8535

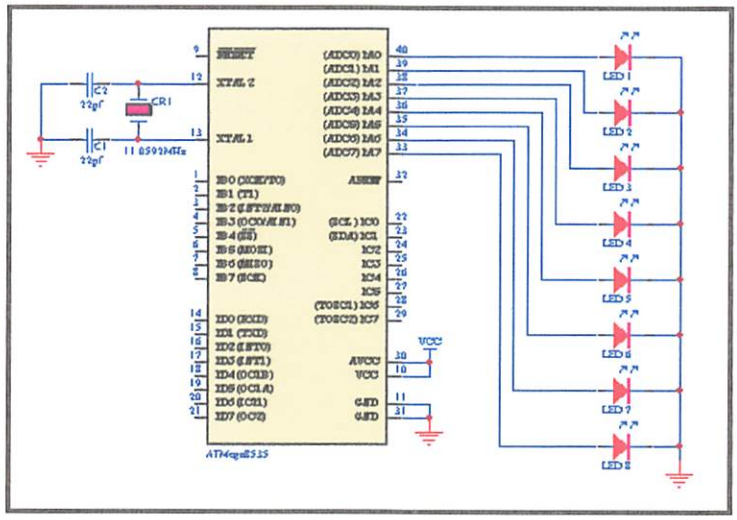
A. Tujuan

Untuk mengetahui apakah rangkaian mikrokontroler ATMega8535 dapat berfungsi dengan baik atau tidak.

B. Peralatan yang digunakan

1. 8 buah LED
2. Mikrokontroller AT Mega8535.
3. Power Supply

C. Rangkaian Pengujian



Gambar 4.2. Rangkaian Pengujian ATMega8535

Langkah –langkah pengujian

1. Merangkai modul rangkaian seperti pada Gambar 4.2
2. Memberikan catu daya pada rangkaian dan mengaktifkannya.
3. Memasukkan program pengujian rangkaian mikrokontroler.
4. Mengamati lampu LED.

D. Data hasil pengujian

Tabel 4-2. Hasil Pengujian Rangkaian ATMega8535

No	PORT A (Biner)	LED 8	LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1
1	0000 0000	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
2	0000 0001	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
3	0000 0010	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
4	0000 0100	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
5	0000 1000	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF

6	0001 0000	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
7	0010 0000	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
8	0100 0000	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
9	1000 0000	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
10	1111 1111	ON	ON	ON	ON	ON	ON	ON	ON

E. Analisa hasil

LED 1- 8 menyala ketika mikrokontroller diberi masukan high (1) dan mati ketika diberi masukan low (0).

F. Kesimpulan

Mikrokontroller sebagai pengedai utama keseluruhan rangkaian berfungsi dengan baik.

4.3. Pengujian Rangkaian LCD

LCD yang digunakan sebagai display adalah jenis LCD M1632 yaitu modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya rendah. Modul tersebut dilengkapi dengan mikrokontroler yang di desain khusus untuk mengendalikan LCD. Mikrokontroler HD44780 buatan Hitachi yang berfungsi sebaga pengendali LCD memiliki CGROM (Character Generator Read Only Memory), dan DDRAM (Display Data Random Access Memory).

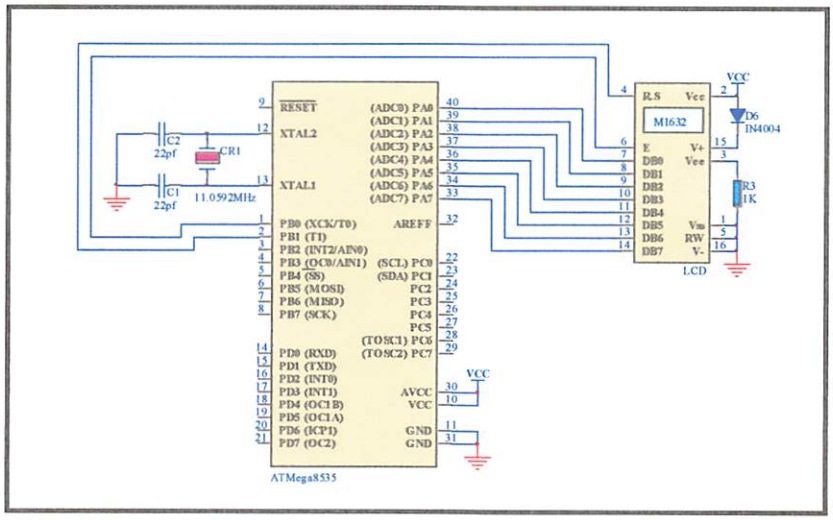
A. Tujuan

Untuk mengetahui apakah rangkaian LCD telah dapat bekerja sesuai dengan yang diharapkan yaitu dapat menampilkan karakter atau string (tulisan) yang diharapkan.

B. Peralatan yang digunakan

1. Rangkaian LCD M1632
2. Rangkaian Mikrokontroller AT Mega8535
3. Power supply

C. Rangkaian Pengujian



Gambar 4.3. Rangkaian Pengujian LCD

Langkah – langkah pengujian:

1. Merangkai modul rangkaian seperti pada Gambar 4.3
2. memberikan catu daya pada rangkaian dan mengaktifkannya.
3. Memasukkan program test LCD pada mikrokontroler ATMega8535

4. Mengamati tampilan pada layar LCD

D. Data hasil pengujian

Gambar 4.4. Foto Hasil Pengujian Rangkaian LCD

E. Analisa hasil

LCD menampilkan tulisan (string) sesuai dengan yang diinginkan.

F. Kesimpulan

LCD berfungsi dengan baik dengan hasil seperti yang diharapkan.

4.4. Pengujian Antarmuka Komunikasi Serial

Interfacing komunikasi serial antara mikrokontroler dengan hp dihubungkan dengan menggunakan kabel data hp. Kabel data yang ada di pasaran pada umumnya telah dilengkapi dengan rangkaian RS232 yang memang oleh vendor pembuat kabel data tersebut, betujuan agar hp dapat digunakan untuk transfer data atau berkomunikasi dengan PC (personal computer) pada level tegangan RS232. Dengan demikikian, mengingat mikrokontroler bekerja dalam level tegangan TTL, maka disini ada dua solusi yang bisa digunakan agar mikrokontroler dapat berkomunikasi dengan hp, yaitu dengan menambahkan rangkaian RS232 pada rangkaian mikrokontroler agar dapat berkomunikasi dalam level tegangan RS232, atau bisa juga dengan memotong ujung kabel data tersebut agar dapat berkomunikasi pada level tegangan TTL. Disini penulis menggunakan level tegangan TTL sehingga kabel data tersebut di potong unjungnya.

A. Tujuan

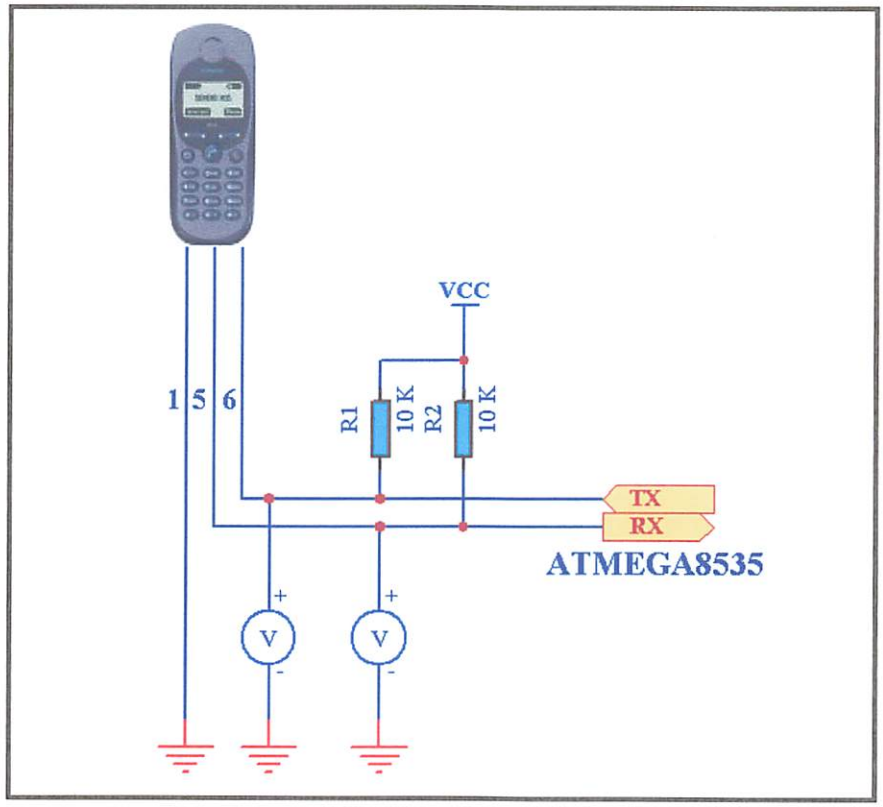
Untuk mengetahui apakah antarmuka serial yang dibangun, telah bekerja dalam level tegangan TTL.

B. Peralatan yang digunakan

1. HP Siemes M35i
2. Kabel data M35i

3. Mikrokontroller AT Mega8535
4. 2 buah multimeter digital tipe DT-830B
5. Power supply

C. Rangkaian Pengujian



Gambar 4.5. Rangkaian Pengujian Antarmuka Serial

Langkah-langkah pengujian:

1. Merangkai modul rangkaian seperti pada Gambar 4.5
2. Menghubungkan rangkaian pada catu daya sebagai sumber tegangan dan mengaktifkannya.
3. Melakukan inisialisasi serial.
4. Mengukur nilai tegangan keluaran pada pin Tx dan Rx pada saat kondisi high dan low.

D. Data hasil pengujian

Tabel 4-3. Hasil Pengujian Antarmuka Serial

Tx (Volt)	**Rx (Volt)**
Kondisi *high*	
4,9	4,9
Kondisi *low*	
0	0

E. Analisa hasil

Pada jalur Tx dan Rx diberikan resistor pull-up sehingga jalur data bersifat open drain, maka dalam kondisi tidak ada data, jalur tersebut dalam keadaan high. Dari hasil pengujian didapatkan logika high pada Rx dan Tx adalah 4,9 Volt dan logika low adalah 0 Volt.

F. Kesimpulan

Keluaran tegangan dari hp sesuai dengan perencanaan, bekerja dalam level tegangan TTL.

4.5. Pengujian Rangkaian Driver Relay

Rangkaian driver relay berfungsi sebagai penguat arus yang diibutuhkan oleh relay agar dapat menghubungkan dan memutuskan aliran arus listrik dengan tegangan AC220V ke beban yang dikontrol. Relay yang digunakan adalah relay DC dengan tegangan koil 12V DC.

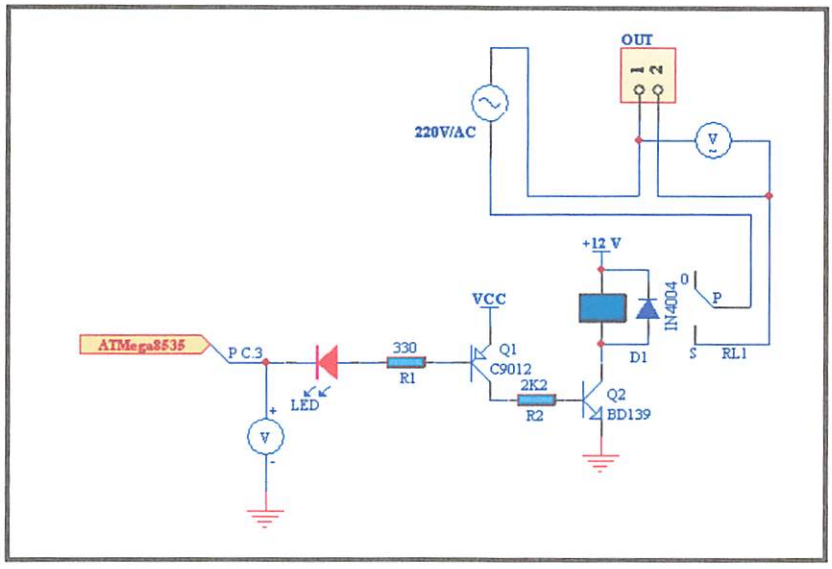
A. Tujuan

Untuk mengetahui kerja relay pada saat bekerja yang berfungsi sebagai saklar pada peralatan listrik.

B. Peralatan yang digunakan

1. Mikrokontroller AT Mega8535
2. Driver relay
3. 2 buah multimeter digital tipe DT-830B
4. Power supply

C. Rangkaian pengujian



Gambar 4.6. Rangkaian Pengujian Driver Relay

Langkah-langkah pengujian :

1. Merangkai modul rangkaian seperti pada Gambar 4.6
2. Menghubungkan catu daya dengan rangkaian sebagai sumber tegangan dan mengaktifkannya.
3. Mengatur multimeter 1 pada DC dengan batas 20V
4. Mengukur tegangan keluaran dari MK ke driver relay.
5. Mengatur multimeter 2 pada AC dengan batas maksimal 250V
6. Mengukur tegangan pada output relay.

D. Data hasil pengujian

Tabel 4-4. Hasil Pengujian Rangkaian Driver Relay

Input Driver Relay (Volt/DC)	**Tegangan pada Output (Volt/AC)**
4,9	221
0	-

E. Analisa hasil

Pengujian rangkaian driver relay secara hardware dilakukan dengan cara memberikan suatu kondisi 'low' (0 Volt) dan 'high' (5 Volt) pada input driver relay, kemudian mengukur tegangan pada koil relay dan tegangan pada output rangkaian driver relay. Dimana relay akan aktif ketika diberi masukan high (5V) dan mati ketika diberi masukan low (0V).

F. Kesimpulan

Relay dapat bekerja dengan baik, yaitu pada masukan high aktif dan pada masukan low mati.

4.6. Pengujian Rangkaian Keseluruhan

A. Tujuan

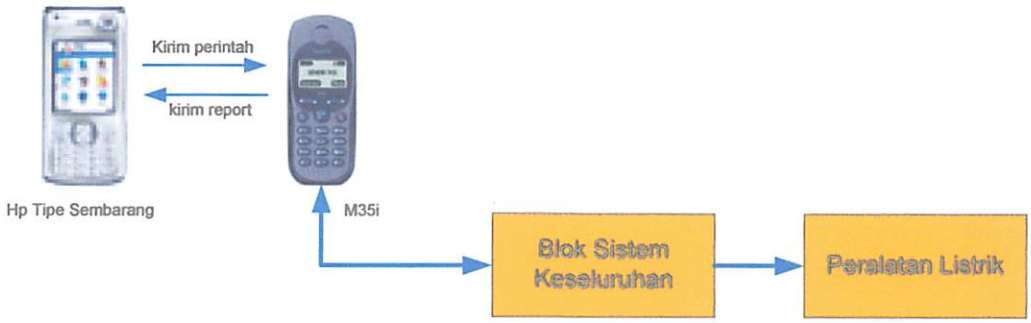
Untuk mengetahui apakah keseluruhan rangkaian telah berfungsi dengan baik dan dapat berfungsi sebagai programmable timer peralatan listrik yang diakses menggunakan SMS.

B. Peralatan yang digunakan

1. HP sembarang tipe pada User
2. HP Siemens M35i pada base terminal
3. Kabel data M35i

4. Rangkaian keseluruhan sistem
    a. Mikrokontroller AT Mega8535
    b. LCD M1632
    c. Driver relay
    d. Power supply
5. Peralatan listrik

C. Blok diagram



Gambar 4.7. Diagram Blok Pengujian Secara Keseluruhan

Langkah-langkah pengujian :

1. Menghubungkan seluruh blok rangkaian menjadi satu system seperti pada Gambar 4.7.
2. Memberikan catu daya sebagai sumber tegangan pada sistem.
3. Menghubungkan sistem MK dengan HP siemens M35i.
4. Memastikan semua bagian telah terhubung dengan benar.
5. Menggunakan Hp tipe sembarang untuk mengirimkan perintah SMS aktif timer peralatan listrik dengan format **Index Output|Mode|Nilai**

Timer, M (menit), **H** (hour/jam). Contoh: "1M5" (tanpa tanda petik) adalah instruksi untuk peralatan listrik 1 aktif dengan timer 5 menit.

6. Mengamati apakah peralatan listrik aktif sampai dengan nilai timer yang telah ditentukan.
7. Menggunakan HP tipe sembarang untuk mengirim perintah SMS menonaktifkan timer peralatan listrik dengan format "1OF" (tanpa tanda petik).
8. Mengamati apakah peralatan listrik telah mati.

D. Data hasil pengujian

Tabel 4-5. Hasil Pengujian Secara Keseluruhan

No	Perintah yang dikirim	Waktu	Peralatan Listrik	
			1	2
1	1M5	09:11	ON	OFF
		09:16	OFF	OFF
2	1M10	09:35	ON	OFF
		09:45	OFF	OFF
3	1M20	10:10	ON	OFF
		10:30	OFF	OFF
4	1M40	10:42	ON	OFF
		11:22	OFF	OFF
5	1M50	12:01	ON	OFF
		12:51	OFF	OFF
6	2M5	14:00	OFF	ON
		14:05	OFF	OFF
7	2M15	14:15	OFF	ON
		14:30	OFF	OFF
8	2M30	14:30	OFF	ON
		15:00	OFF	OFF
9	2H1	15:05	OFF	ON
		16:05	OFF	OFF
10	2H3	16:23	OFF	ON
		19:23	OFF	OFF

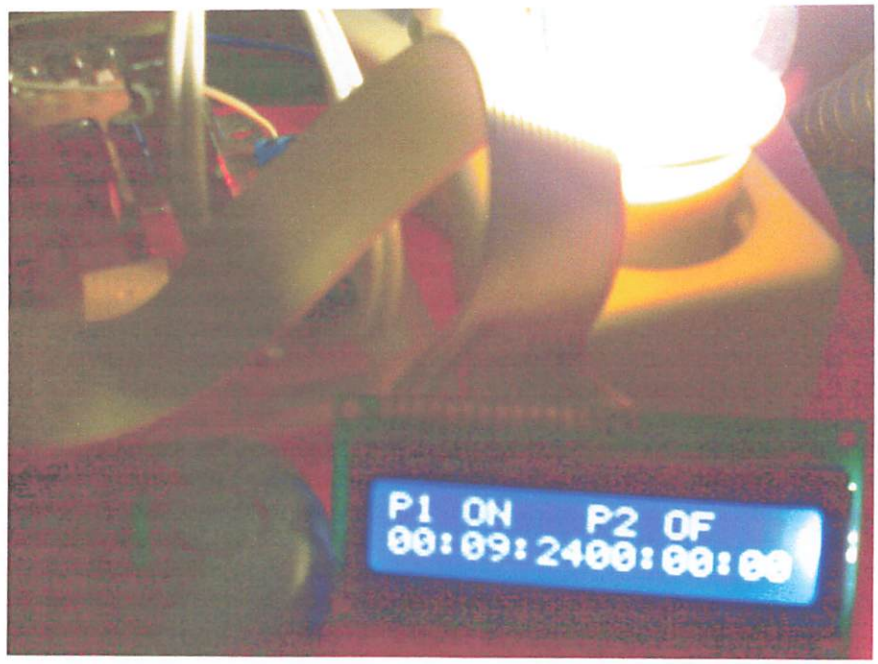
E. Analisa hasil

Sistem merespon ketika ada SMS perintah pengaktifan dengan mengaktifkan timer peralatan listrik sesuai dengan waktu yang diinginkan dan kemudian mengirimkan report ke HP pengirim sebagai tanda peralatan listrik telah aktif. Ketika nilai timer telah tercapai maka system akan mematikan peralatan listrik.

F. Kesimpulan

Keseluruhan rangkaian berfungsi dengan baik dan dapat berfungsi sebagai programmable timer peralatan listrik yang diakses menggunakan SMS.

Gambar 4.8. Foto Alat Pada Saat Peralatan Listrik OFF



Gambar 4.9. Foto alat pada saat timer peralatan listrik 1 aktif 10 menit

Gambar 4.10. Foto alat pada saat timer peralatan listrik 1 dan 2 aktif

BAB V

PENUTUP

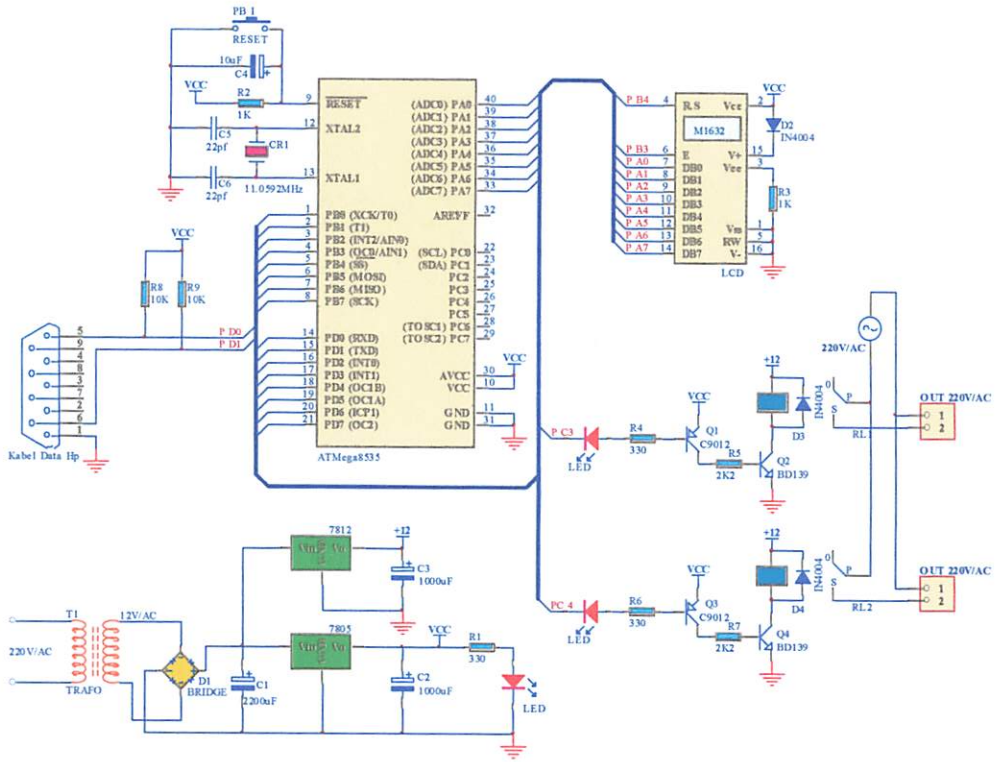
5.1. Kesimpulan

Kesimpulan yang didapat adalah sebagai berikut :

1. Alat ini dapat bekerja dengan baik mengontrol peralatan listrik dalam dua mode yaitu baik mode tanpa timer maupun pada mode menggunakan timer.
2. Untuk memastikan sistem komunikasi serial antara telepon seluler dengan mikrokontroller AVR ATmega8535 dapat berfungsi, pin-pin transmitter dan receiver harus bertegangan logika 1.
3. Inisialisasi LCD berhasil apabila LCD dapat menampilkan data yang dikirim melalui pin-pin data.
4. Data yang dikirimkan telepon seluler sebagai tanggapan atas instruksi yang diterima, dikirim menggunakan format PDU (Protocol Data Unit) dalam kode ASCII. Untuk dapat diambil informasinya diperlukan pengubahan dari kode Hex-Ascii ke kode Heksa, diteruskan dari kode Heksa ke kode ASCII.
5. Sistem SMS sangat tergantung pada keadaan jaringan, dari penerimaan sinyal pada handphone user maupun pada handphone base terminal.

5.2. Saran

Beberapa kekurangan alat ini adalah harus membayar setiap kali mengirim sms, juga kartu memiliki masa aktif dan kadang SMS terlambat atau malah tidak sampai, masalah ini dapat diatasi dengan menggunakan kartu yang sejenis, sehingga SMS dapat segera sampai.



Title	PROGRAMMABLE TIMER PERALATAN LISTRIK DI AKSES VIA SMS	
Size A4	RANU SETYOBUDI 0017062 TEKNIK ELEKTRONIKA S-1 ITN MALANG 2009	ITN TUGAS AKHIR

LISTING PROGRAM

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Listing Program Programmable Timer
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.include"m8535def.inc"
.def    P1 =R2
.def    P2 =R3
.def    T1 =R4
.def    T2 =R5
.def    H1 =R6
.def    M1 =R7
.def    S1 =R8
.def    H2 =R9
.def    M2 =R12
.def    S2 =R13
.def    A1 =R14
.def    U7S =R25
.def    BCDH =R24
.def    BCDL =R23
.def    tmp =r16
.def    txbyte =r17
.def    rxbyte =r18
.def    tmp_1 =r19
.equ    fclock  =11059200
.equ    baud_rate =19200
.equ    ubbr_value =(fclock/(16*baud_rate))-1
.equ timer_value1=0xD5D0
.equ timer_value = 0xffca
.equ timer_value100 = 0xfbaf
.equ RS_LCD=7
.equ E_LCD=6
.equ RW_LCD=2
.EQU DISPCLR=0b00000001
.EQU FUNCSET=0b00111000
.EQU ENTRMOD=0b00000110
.EQU DISPON=0b00001100
.EQU PLCD=PORTA
.EQU SDA = PC1
.EQU SCL= PC0
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Awal Program Programmable Timer
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.org 0x0000
rjmp    main
;.org 0x00B
;rjmp   usart_rxc
main:
```

```
ldi     r16,low(RAMEND)
out     SPL,r16
ldi     r16,high(RAMEND)
out     SPH,r16
ldi     r16,0xff
out     ddra,r16
ldi     r19,0x00
out     porta,r19       ;sebagai output
out     ddrb,r16
out     portb,r16       ;sebagai output data ke LCD
out     ddrc,r16
out     portc,r19       ;sebagai output E, RS,R/W
ldi     tmp,0b01000000
out     ddrd,tmp
out     portd,tmp
rcall   init_usart
rcall   Init_LCD1
rcall   delay1d
rcall   delay1d
rcall   reset_all
rcall   Iklan
;rcall  kirim_csms
;rjmp   seting
;cbi    portb,3
;cbi    portd,6
mulai:
rcall   dis_play
rcall   cek_timer
rcall   delay1d
rcall   delay1d
rcall   delay1d
rcall   delay1d
rcall   delay1d
;sbi    portb,3
rcall   delay1d
rcall   delay1d
rcall   delay1d
rcall   delay1d
rcall   delay1d
rcall   cek_sms
rjmp    Mulai
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; SUBRUTIN SET KIRIM SMS
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;KIRIM AT+CMGS (MENGIRIM SMS)
;--------------------------
```

```
tes_report:
rcall   kirim_cmgs
ldi     zl,low(2*msgT1_ON)
ldi     zh,high(2*msgT1_ON)
S_T1_ON:
lpm
mov     txbyte,r0
cpi     txbyte,0
breq    Out_T1_ON
rcall   usart_tx
adiw    zh:zl,1
rjmp    S_T1_ON

out_T1_ON:
rcall   kirim1A
ret
kirim1A:
ldi     txbyte,$1A
rcall   usart_tx
ret

kirim_cmgs:
ldi     zl,low(2*msg_cmgs)
ldi     zh,high(2*msg_cmgs)

cmgs:
lpm
mov     txbyte,r0
cpi     txbyte,0
breq    tunggu
rcall   usart_tx
adiw zh:zl,1
rjmp    cmgs
msg_cmgs:
.db     "AT+CMGS=27",13,10,0      ;(13+jml karakter)

tunggu:
rcall   usart_rx
cpi     rxbyte,'>'
breq    kirim_header
rjmp    tunggu

kirim_header:
ldi     zl,low(2*msgheader)
ldi     zh,high(2*msgheader)
```

```
header:
lpm
mov    txbyte,r0
cpi    txbyte,0
breq   kirim_pesan
rcall  usart_tx
adiw zh:zl,1
rjmp   header
msgheader:
.db    "07912658050000f001000d912658461993 58f90000",0

kirim_pesan:
ret
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; SUBRUTIN SET TERIMA SMS
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;-------------------------
cek_sms:
periksa_sms:
rcall  KIRIM_CMGR
mov    tmp,A1
cpi    tmp,1
breq   pesanterima
ret

pesanterima:
rcall  usart_rx
rcall  usart_rx
cpi    rxbyte,'7'
brne   cek_06
ldi    tmp,0
loop_07:
rcall  usart_rx
cpi    tmp,15
breq   cek_nomor
inc    tmp
rjmp   loop_07

cek_06:
cpi    rxbyte,'6'
brne   out_cek_SMSC
ldi    tmp,0
loop_06:
rcall  usart_rx
cpi    tmp,13
breq   cek_nomor
```

```
inc     tmp
rjmp    loop_06

out_cek_SMSC:
rcall   kirim_cmgd
ret

cek_nomor:
ldi     tmp,0
loop_cek_N:
rcall   usart_rx
cpi     tmp,21
breq    skipwaktu
inc     tmp
rjmp    loop_cek_N

cek_nomor1:
ldi     zl,low(2*msg_user)
ldi     zh,high(2*msg_user)

loadperintah:
lpm
mov     r29,r0
cpi     r29,0           ;apakah pembanding sudah nol
breq    skipwaktu       ;jika sudah lompat ke skip waktu
rcall   usart_rx
cp      rxbyte,r0       ;cocokkan karakter sms
brne    bukanuser       ;jika tidak cocok bukan hp user
adiw    zh:zl,1
rjmp    loadperintah

msg_user:
.db     "0D91265846199358F90000",0      ;sms terima

bukanuser:
rcall   kirim_cmgd              ; kirim AT+CMGD=1
ret

skipwaktu:
ldi             r16,13          ;skip 16 bit data waktu
loopskip:
rcall   usart_rx
cpi     r16,0
breq    end_skip
dec     r16
rjmp    loopskip
```

```
end_skip:
rcall   usart_rx
cpi     rxbyte,'0'
breq    cek_03
rjmp    kirim_false

cek_03:
rcall   usart_rx
cpi     rxbyte,'3'
brne    cek_04
clr     r27
ldi     r26,$60
ldi     tmp,0
Loop_cek03:
rcall   usart_rx
st      x+,rxbyte
cpi     tmp,5
breq    space_1
inc     tmp
rjmp    loop_cek03

cek_04:
cpi     rxbyte,'4'
brne    kirim_false
clr     r27
ldi     r26,$60
ldi     tmp,0

Loop_cek04:
rcall   usart_rx
st      x+,rxbyte
cpi     tmp,7
breq    space_2
inc     tmp
rjmp    loop_cek04

kirim_false:
rcall   kirim_cmgd
ret
space_2:
rjmp    to_space_2
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
space_1:
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
;*************************************************************
cek_T1_ON:
;*************************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msgT1_ON)
ldi     zh,high(2*msgT1_ON)
adiw    zh:zl,2

Loop_1_ON:
lpm
mov     r29,r0
cpi     r29,0
breq    T1_ON
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_T1_OF
adiw    zh:zl,1
rjmp    Loop_1_ON

T1_ON:
ldi     tmp,1
mov     P1,tmp
rcall   kirim_cmgd
cbi     portb,3
ret
;*************************************************************
cek_T1_OF:
;*************************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msgT1_OF)
ldi     zh,high(2*msgT1_OF)
adiw    zh:zl,2

Loop_1_OF:
lpm
mov     r29,r0
cpi     r29,0
breq    T1_OF
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M5
adiw    zh:zl,1
rjmp    Loop_1_OF
```

```
T1_OF:
rcall   reset_T1
rcall   kirim_cmgd
sbi     portb,3
ret
;*********************************************************
cek_1M5:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M5)
ldi     zh,high(2*msg_1M5)
adiw    zh:zl,2

Loop_1M5:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M5
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1H1
adiw    zh:zl,1
rjmp    Loop_1M5

IN_1M5:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,5
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1H1:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1H1)
ldi     zh,high(2*msg_1H1)
adiw    zh:zl,2

Loop_1H1:
lpm
mov     r29,r0
```

```
cpi     r29,0
breq    IN_1H1
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1H2
adiw    zh:zl,1
rjmp    Loop_1H1

IN_1H1:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,1
mov     H1,tmp
rcall   kirim_cmgd
ret

;**************************
cek_1H2:
;**************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1H2)
ldi     zh,high(2*msg_1H2)
adiw    zh:zl,2

Loop_1H2:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1H2
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1H3
adiw    zh:zl,1
rjmp    Loop_1H2

IN_1H2:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,2
mov     H1,tmp
rcall   kirim_cmgd
ret
```

```
;*********************************************************
cek_1H3:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1H3)
ldi     zh,high(2*msg_1H3)
adiw    zh:zl,2

Loop_1H3:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1H3
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_T2_ON
adiw    zh:zl,1
rjmp    Loop_1H3

IN_1H3:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,3
mov     H1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_T2_ON:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msgT2_ON)
ldi     zh,high(2*msgT2_ON)
adiw    zh:zl,2

Loop_2_ON:
lpm
mov     r29,r0
cpi     r29,0
breq    T2_ON
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_T2_OF
```

```
adiw   zh:zl,1
rjmp   Loop_2_ON

T2_ON:
ldi    tmp,1
mov    P2,tmp
rcall  kirim_cmgd
cbi    portd,6
ret

;*********************************************************
cek_T2_OF:
;*********************************************************
clr    r27
ldi    r26,$60
ldi    zl,low(2*msgT2_OF)
ldi    zh,high(2*msgT2_OF)
adiw   zh:zl,2

Loop_2_OF:
lpm
mov    r29,r0
cpi    r29,0
breq   T2_OF
ld     tmp_1,x+
cp     tmp_1,r0
brne   cek_2M5
adiw   zh:zl;1
rjmp   Loop_2_OF

T2_OF:
rcall  reset_T2
rcall  kirim_cmgd
sbi    portd,6
ret

;*********************************************************
cek_2M5:
;*********************************************************
clr    r27
ldi    r26,$60
ldi    zl,low(2*msg_2M5)
ldi    zh,high(2*msg_2M5)
adiw   zh:zl,2

Loop_2M5:
```

```
lpm
mov   r29,r0
cpi   r29,0
breq  IN_2M5
ld    tmp_1,x+
cp    tmp_1,r0
brne  cek_2H1
adiw  zh:zl,1
rjmp  Loop_2M5

IN_2M5:
ldi   tmp,1          ;register tanda timer 1 On
mov   T2,tmp
mov   P2,tmp
ldi   tmp,5
mov   m2,tmp
rcall kirim_cmgd
ret

;*********************************************************
cek_2H1:
;*********************************************************
clr   r27
ldi   r26,$60
ldi   zl,low(2*msg_2H1)
ldi   zh,high(2*msg_2H1)
adiw  zh:zl,2

Loop_2H1:
lpm
mov   r29,r0
cpi   r29,0
breq  IN_2H1
ld    tmp_1,x+
cp    tmp_1,r0
brne  cek_2H2
adiw  zh:zl,1
rjmp  Loop_2H1

IN_2H1:
ldi   tmp,1          ;register tanda timer 1 On
mov   T2,tmp
mov   P2,tmp
ldi   tmp,1
mov   H2,tmp
rcall kirim_cmgd
```

```
ret

;*********************************************************
cek_2H2:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2H2)
ldi     zh,high(2*msg_2H2)
adiw    zh:zl,2

Loop_2H2:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2H2
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2H3
adiw    zh:zl,1
rjmp    Loop_2H2

IN_2H2:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,2
mov     H2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2H3:
;*********************************************************
clr     r27
ldi             r26,$60
ldi             zl,low(2*msg_2H3)
ldi             zh,high(2*msg_2H3)
adiw    zh:zl,2

Loop_2H3:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2H3
ld      tmp_1,x+
```

```
cp      tmp_1,r0
brne    out_space1
adiw    zh:zl,1
rjmp    Loop_2H3

IN_2H3:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,3
mov     H2,tmp
rcall   kirim_cmgd
ret

out_space1:
rcall   kirim_cmgd
ret
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
to_space_2:
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;**********************************************************
cek_1M10:
;**********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M10)
ldi     zh,high(2*msg_1M10)
adiw    zh:zl,2
Loop_1M10:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M10
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M15
adiw    zh:zl,1
rjmp    Loop_1M10

IN_1M10:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,10
mov     m1,tmp
rcall   kirim_cmgd
```

```
ret

;*********************************************************
cek_1M15:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M15)
ldi     zh,high(2*msg_1M15)
adiw    zh:zl,2

Loop_1M15:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M15
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M20
adiw    zh:zl,1
rjmp    Loop_1M15

IN_1M15:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,15
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M20:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M20)
ldi     zh,high(2*msg_1M20)
adiw    zh:zl,2

Loop_1M20:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M20
ld      tmp_1,x+
```

```
cp      tmp_1,r0
brne    cek_1M25
adiw    zh:zl,1
rjmp    Loop_1M20

IN_1M20:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,20
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M25:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M25)
ldi     zh,high(2*msg_1M25)
adiw    zh:zl,2

Loop_1M25:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M25
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M30
adiw    zh:zl,1
rjmp    Loop_1M25

IN_1M25:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,25
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M30:
;*********************************************************
```

```
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M30)
ldi     zh,high(2*msg_1M30)
adiw    zh:zl,2

Loop_1M30:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M30
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M35
adiw    zh:zl,1
rjmp    Loop_1M30

IN_1M30:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,30
mov     m1,tmp
rcall   kirim_cmgd
ret

;**************************
cek_1M35:
;**************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M35)
ldi     zh,high(2*msg_1M35)
adiw    zh:zl,2

Loop_1M35:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M35
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M40
adiw    zh:zl,1
rjmp    Loop_1M35
```

```
IN_1M35:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,35
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M40:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M40)
ldi     zh,high(2*msg_1M40)
adiw    zh:zl,2

Loop_1M40:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M40
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_1M45
adiw    zh:zl;1
rjmp    Loop_1M40

IN_1M40:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,40
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M45:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M45)
ldi     zh,high(2*msg_1M45)
```

```
adiw   zh:zl,2

Loop_1M45:
lpm
mov    r29,r0
cpi    r29,0
breq   IN_1M45
ld     tmp_1,x+
cp     tmp_1,r0
brne   cek_1M50
adiw   zh:zl,1
rjmp   Loop_1M45

IN_1M45:
ldi    tmp,1           ;register tanda timer 1 On
mov    T1,tmp
mov    P1,tmp
ldi    tmp,45
mov    m1,tmp
rcall  kirim_cmgd
ret

;*********************************************************
cek_1M50:
;*********************************************************
clr    r27
ldi    r26,$60
ldi    zl,low(2*msg_1M50)
ldi    zh,high(2*msg_1M50)
adiw   zh:zl,2

Loop_1M50:
lpm
mov    r29,r0
cpi    r29,0
breq   IN_1M50
ld     tmp_1,x+
cp     tmp_1,r0
brne   cek_1M55
adiw   zh:zl,1
rjmp   Loop_1M50

IN_1M50:
ldi    tmp,1           ;register tanda timer 1 On
mov    T1,tmp
mov    P1,tmp
```

```
ldi     tmp,50
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_1M55:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_1M55)
ldi     zh,high(2*msg_1M55)
adiw    zh:zl,2

Loop_1M55:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_1M55
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M10
adiw    zh:zl,1
rjmp    Loop_1M55

IN_1M55:
ldi     tmp,1           ;register tanda timer 1 On
mov     T1,tmp
mov     P1,tmp
ldi     tmp,55
mov     m1,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M10:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M10)
ldi     zh,high(2*msg_2M10)
adiw    zh:zl,2

Loop_2M10:
lpm
mov     r29,r0
```

```
cpi     r29,0
breq    IN_2M10
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M15
adiw    zh:zl,1
rjmp    Loop_2M10

IN_2M10:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,10
mov     m2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M15:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M15)
ldi     zh,high(2*msg_2M15)
adiw    zh:zl,2

Loop_2M15:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M15
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M20
adiw    zh:zl,1
rjmp    Loop_2M15

IN_2M15:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,15
mov     m2,tmp
rcall   kirim_cmgd
ret
```

```
;*********************************************************
cek_2M20:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M20)
ldi     zh,high(2*msg_2M20)
adiw    zh:zl,2

Loop_2M20:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M20
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M25
adiw    zh:zl,1
rjmp    Loop_2M20

IN_2M20:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,20
mov     m2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M25:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M25)
ldi     zh,high(2*msg_2M25)
adiw    zh:zl,2

Loop_2M25:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M25
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M30
```

```
adiw    zh:zl,1
rjmp    Loop_2M25

IN_2M25:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,25
mov     m2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M30:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M30)
ldi     zh,high(2*msg_2M30)
adiw    zh:zl,2

Loop_2M30:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M30
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M35
adiw    zh:zl,1
rjmp    Loop_2M30

IN_2M30:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,30
mov     m2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M35:
;*********************************************************
clr     r27
ldi     r26,$60
```

```
ldi     zl,low(2*msg_2M35)
ldi     zh,high(2*msg_2M35)
adiw    zh:zl,2

Loop_2M35:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M35
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M40
adiw    zh:zl,1
rjmp    Loop_2M35

IN_2M35:
ldi     tmp,1           ;register tanda timer 1 On
mov     T2,tmp
mov     P2,tmp
ldi     tmp,35
mov     m2,tmp
rcall   kirim_cmgd
ret

;*********************************************************
cek_2M40:
;*********************************************************
clr     r27
ldi     r26,$60
ldi     zl,low(2*msg_2M40)
ldi     zh,high(2*msg_2M40)
adiw    zh:zl,2

Loop_2M40:
lpm
mov     r29,r0
cpi     r29,0
breq    IN_2M40
ld      tmp_1,x+
cp      tmp_1,r0
brne    cek_2M45
adiw    zh:zl,1
rjmp    Loop_2M40

IN_2M40:
ldi     tmp,1           ;register tanda timer 1 On
```

```
mov   T2,tmp
mov   P2,tmp
ldi   tmp,40
mov   m2,tmp
rcall kirim_cmgd
ret

;*********************************************************
cek_2M45:
;*********************************************************
clr   r27
ldi   r26,$60
ldi   zl,low(2*msg_2M45)
ldi   zh,high(2*msg_2M45)
adiw  zh:zl,2

Loop_2M45:
lpm
mov   r29,r0
cpi   r29,0
breq  IN_2M45
ld    tmp_1,x+
cp    tmp_1,r0
brne  cek_2M50
adiw  zh:zl,1
rjmp  Loop_2M45

IN_2M45:
ldi   tmp,1           ;register tanda timer 1 On
mov   T2,tmp
mov   P2,tmp
ldi   tmp,45
mov   m2,tmp
rcall kirim_cmgd
ret

;*********************************************************
cek_2M50:
;*********************************************************
clr   r27
ldi   r26,$60
ldi   zl,low(2*msg_2M50)
ldi   zh,high(2*msg_2M50)
adiw  zh:zl,2

Loop_2M50:
```

```
lpm
mov   r29,r0
cpi   r29,0
breq  IN_2M50
ld    tmp_1,x+
cp    tmp_1,r0
brne  cek_2M55
adiw  zh:zl,1
rjmp  Loop_2M50

IN_2M50:
ldi   tmp,1          ;register tanda timer 1 On
mov   T2,tmp
mov   P2,tmp
ldi   tmp,50
mov   m2,tmp
rcall kirim_cmgd
ret

;*********************************************************
cek_2M55:
;*********************************************************
clr   r27
ldi   r26,$60
ldi   zl,low(2*msg_2M55)
ldi   zh,high(2*msg_2M55)
adiw  zh:zl,2

Loop_2M55:
lpm
mov   r29,r0
cpi   r29,0
breq  IN_2M55
ld    tmp_1,x+
cp    tmp_1,r0
brne  out_space2
adiw  zh:zl,1
rjmp  Loop_2M55

IN_2M55:
ldi   tmp,1          ;register tanda timer 1 On
mov   T2,tmp
mov   P2,tmp
ldi   tmp,55
mov   m2,tmp
rcall kirim_cmgd
```

```
ret

out_space2:
rcall   kirim_cmgd
ret

msgT1_ON:
.db     "03B1A713",0        ;1ON

msgT1_OF:
.db     "03B1A711",0        ;1OF

msg_1M1:
.db     "03B1660C",0;1M1

msg_1M2:
.db     "03B1A60C",0        ;1M2

msg_1M3:
.db     "03B1E60C",0        ;1M3

msg_1M4:
.db     "03B1260D",0;1M4

msg_1M5:
.db     "03B1660D",0;1M5

msg_1M6:
.db     "03B1A60D",0        ;1M6

msg_1M7:
.db     "03B1E60D",0        ;1M7

msg_1M8:
.db     "03B1260E",0;1M8

msg_1M9:
.db     "03B1660E",0;1M9

msg_1M10:
.db     "04B1660C06",0      ;1M10

msg_1M15:
.db     "04B166AC06",0      ;1M15

msg_1M20:
```

```
.db     "04B1A60C06",0      ;1M20

msg_1M25:
.db     "04B1A6AC06",0      ;1M25

msg_1M30:
.db     "04B1E60C06",0      ;1M30

msg_1M35:
.db     "04B1E6AC06",0      ;1M35

msg_1M40:
.db     "04B1260D06",0      ;1M40

msg_1M45:
.db     "04B126AD06",0      ;1M45

msg_1M50:
.db     "04B1660D06",0      ;1M50

msg_1M55:
.db     "04B166AD06",0      ;1M55

msg_1H1:
.db     "0331640C",0 ;1H1

msg_1H2:
.db     "0331A40C",0        ;1H2

msg_1H3:
.db     "0331E40C",0        ;1H3

;=============================
msgT2_ON:
.db     "03B2A713",0        ;2ON

msgT2_OF:
.db     "03B2A711",0        ;2OF

msg_2M1:
.db     "03B2660C",0;2M1

msg_2M2:
.db     "03B2A60C",0        ;2M2

msg_2M3:
```

```
.db     "03B2E60C",0          ;2M3

msg_2M4:
.db     "03B2260D",0;2M4

msg_2M5:
.db     "03B2660D",0;2M5

msg_2M6:
.db     "03B2A60D",0          ;2M6

msg_2M7:
.db     "03B2E60D",0          ;2M7

msg_2M8:
.db     "03B2260E",0 ;2M8

msg_2M9:
.db     "03B2660E",0 ;2M9

msg_2M10:
.db     "04B2660C06",0        ;2M10

msg_2M15:
.db     "04B266AC06",0        ;2M15


msg_2M20:
.db     "04B2A60C06",0        ;2M20

msg_2M25:
.db     "04B2A6AC06",0        ;2M25

msg_2M30:
.db     "04B2E60C06",0        ;2M30

msg_2M35:
.db     "04B2E6AC06",0        ;2M35

msg_2M40:
.db     "04B2260D06",0        ;2M40

msg_2M45:
.db     "04B226AD06",0        ;2M45

msg_2M50:
```

```
.db     "04B2660D06",0      ;2M50

msg_2M55:
.db     "04B266AD06",0      ;2M55

msg_2H1:
.db     "0332640C",0 ;2H1

msg_2H2:
.db     "0332A40C",0;2H2

msg_2H3:
.db     "0332E40C",0 ;2H3

R_False:
.db     "05C620735A04",0

;*********************************************************
;KIRIM AT+CMGR=1
;*********************************************************
KIRIM_CMGR:
ldi     zl,low(2*msgcmgr)
ldi     zh,high(2*msgcmgr)

CMGR:
lpm
mov     txbyte,r0
cpi     txbyte,0
breq    tunggunol1
rcall   usart_tx
adiw    zh:zl,1
rjmp    CMGR

msgcmgr:
.db     "at+CMGR=1",13,10,0         ;(13+jml karakter)

tunggunol1:
loop_skip:

rcall   usart_rx
cpi     rxbyte,':'
brne    loop_skip
rcall   usart_rx
rcall   usart_rx
cpi     rxbyte,'0'
brne    menuju_data
```

```
ret

menuju_data:
rcall   usart_rx
cpi     rxbyte,$0D
brne    menuju_data
rcall   usart_rx        ;setelah $0D masih ada data satu lagi
ldi     tmp,1
mov     A1,tmp
ret

;*********************************************************
;KIRIM AT+CMGD=1
;*********************************************************
kirim_cmgd:
rcall   delay1d
rcall   delay1d
rcall   delay1d
ldi     zl,low(2*msgcmgd)
ldi     zh,high(2*msgcmgd)
load10:
lpm
mov     txbyte,r0
cpi     txbyte,0
breq    tunggu_cmgd
rcall   usart_tx
adiw    zh:zl,1
rjmp    load10

msgcmgd:
.db     "at+cmgd=1",13,10,0

tunggu_cmgd:
rcall   usart_rx
cpi     rxbyte,$4B
breq    kembali_cmgd
rjmp    tunggu_cmgd

kembali_cmgd:
ldi     tmp,0
mov     A1,tmp
rcall   delay1d
ret
```

```
;*********************************************************
cek_timer:
;*********************************************************
cpi     U7S,9
breq    cek_sms_bentar

cek_status_t:
mov     tmp,t1
cpi     tmp,1
brne    cek_t2
cbi     portb,3

cek_t1:
rcall   delay1d
inc     U7S
mov     tmp,s1
cpi     tmp,0
brne    dec_s1

mov     tmp,m1
cpi     tmp,0
brne    dec_m1

mov     tmp,h1
cpi     tmp,0
brne    dec_h1

rcall   reset_timer_1
rjmp    cek_t2

cek_sms_bentar:
ldi     U7S,0
rcall   cek_sms
rjmp    cek_status_t

dec_s1:
dec     s1
rjmp    cek_t2

dec_m1:
dec     m1
ldi     tmp,59
mov     s1,tmp
rjmp    cek_t2

dec_h1:
```

```
dec     h1
ldi     tmp,59
mov     m1,tmp
rjmp    cek_t2

reset_timer_1:
ldi     tmp,0
mov     p1,tmp
mov     t1,tmp
mov     s1,tmp
mov     m1,tmp
mov     h1,tmp
sbi     portb,3
ret

cek_t2:
mov     tmp,t2
cpi     tmp,1
brne    cek_t3
sbrs    t1,0
rcall   delay1d
sbrs    t1,0
inc     U7S
cbi     portd,6

mov     tmp,s2
cpi     tmp,0
brne    dec_s2

mov     tmp,m2
cpi     tmp,0
brne    dec_m2

mov     tmp,h2
cpi     tmp,0
brne    dec_h2
rcall   reset_timer_2
rjmp    cek_t3

dec_s2:
dec     s2
rjmp    cek_t3

dec_m2:
dec     m2
ldi     tmp,59
```

```
mov    s2,tmp
rjmp   cek_t3

dec_h2:
dec    h2
ldi    tmp,59
mov    m2,tmp
rjmp   cek_t3

reset_timer_2:
ldi    tmp,0
mov    p2,tmp
mov    t2,tmp
mov    s2,tmp
mov    m2,tmp
mov    h2,tmp
sbi    portd,6
ret


cek_t3:
rcall  dis_play
mov    tmp,t1
cpi    tmp,0
breq   cek_st2
rjmp   cek_timer

cek_st2:
mov    tmp,t2
cpi    tmp,0
breq   out_cek
rjmp   cek_timer

out_cek:
ldi    U7S,0
ret

dis_play:
ldi    tmp,0x80
rcall  Kirim_Perintah
rcall  P_

ldi    tmp,0x81
rcall  Kirim_Perintah
ldi    tmp,'1'
rcall  Kirim_karakter
```

```
ldi     tmp,0x83
rcall   Kirim_Perintah
rcall   O_

mov     tmp,p1
cpi     tmp,0
brne    P1_on_call
rcall   P1_off
rjmp    terus_1

P1_on_call:
rcall   P1_on

terus_1:
ldi     tmp,0x88
rcall   Kirim_Perintah
rcall   P_

ldi     tmp,0x89
rcall   Kirim_Perintah
ldi     tmp,'2'
rcall   Kirim_karakter

ldi     tmp,0x8B
rcall   Kirim_Perintah
rcall   O_

mov     tmp,p2
cpi     tmp,0
brne    P2_on_call
rcall   P2_off
rjmp    terus_2

P2_on_call:
rcall   P2_on

terus_2:
;----------------------- Hour 1
mov     tmp,h1
rcall   bin2bcd8
ldi     tmp,0xC0
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter
```

```
ldi     tmp,0xC1
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter

ldi     tmp,0xc2
rcall   Kirim_Perintah
ldi     tmp,':'
rcall   Kirim_karakter


;----------------------- Minute 1
mov     tmp,m1
rcall   bin2bcd8
ldi     tmp,0xC3
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter

ldi     tmp,0xC4
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter

ldi     tmp,0xc5
rcall   Kirim_Perintah
ldi     tmp,':'
rcall   Kirim_karakter

;----------------------- Secon 1
mov     tmp,s1
rcall   bin2bcd8
ldi     tmp,0xC6
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter

ldi     tmp,0xC7
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter

;----------------------- Hour 2
mov     tmp,h2
rcall   bin2bcd8
```

```
ldi     tmp,0xC8
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter

ldi     tmp,0xC9
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter

ldi     tmp,0xca
rcall   Kirim_Perintah
ldi     tmp,':'
rcall   Kirim_karakter

;----------------------- Minute 2
mov     tmp,m2
rcall   bin2bcd8
ldi     tmp,0xCb
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter

ldi     tmp,0xCc
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter

ldi     tmp,0xcd
rcall   Kirim_Perintah
ldi     tmp,':'
rcall   Kirim_karakter

;----------------------- Secon 2
mov     tmp,s2
rcall   bin2bcd8
ldi     tmp,0xCe
rcall   Kirim_Perintah
mov     tmp,BCDH
rcall   Kirim_karakter

ldi     tmp,0xCf
rcall   Kirim_Perintah
mov     tmp,BCDL
rcall   Kirim_Karakter
ret
```

```
;*********************************************************
bin2bcd8:
;*********************************************************
        clr     BCDH            ;clear result MSD
bBCD8_1:
        subi    tmp,10          ;input = input - 10
        brcs    bBCD8_2         ;abort if carry set
        inc     BCDH            ;inc MSD
        rjmp    bBCD8_1         ;loop again

bBCD8_2:
        subi    tmp,-10         ;compensate extra subtraction
        ldi     tmp_1,$30
        add     tmp,tmp_1
        mov     BCDL,tmp
        add     BCDH,tmp_1
        ret

P1_off:
ldi     tmp,0x84
rcall   Kirim_Perintah
rcall   F_
ret

P1_on:
ldi     tmp,0x84
rcall   Kirim_Perintah
rcall   N_
ret

P2_off:
ldi     tmp,0x8C
rcall   Kirim_Perintah
rcall   F_
ret

P2_on:
ldi     tmp,0x8C
rcall   Kirim_Perintah
rcall   N_
ret

O_:
ldi     tmp,'O'
rcall   Kirim_karakter
ret
```

```
P_:
ldi     tmp,'P'
rcall   Kirim_karakter
ret

F_:
ldi     tmp,'F'
rcall   Kirim_karakter
ret

N_:
ldi     tmp,'N'
rcall   Kirim_karakter
ret

;*********************************************************
Iklan:
;*********************************************************
ldi     ZL,low(2*TEXT_1)
ldi     ZH,high(2*TEXT_1)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_2)
ldi     ZH,high(2*TEXT_2)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_3)
ldi     ZH,high(2*TEXT_3)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_4)
ldi     ZH,high(2*TEXT_4)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_5)
ldi     ZH,high(2*TEXT_5)
rcall   Baris1
```

```
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_6)
ldi     ZH,high(2*TEXT_6)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_7)
ldi     ZH,high(2*TEXT_7)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_8)
ldi     ZH,high(2*TEXT_8)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_9)
ldi     ZH,high(2*TEXT_9)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_A)
ldi     ZH,high(2*TEXT_A)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_B)
ldi     ZH,high(2*TEXT_B)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_C)
ldi     ZH,high(2*TEXT_C)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     ZL,low(2*TEXT_D)
```

```
ldi     ZH,high(2*TEXT_D)
rcall   Baris1
rcall   KirimPesan_LCD
ldi     ZL,low(2*TEXT_E)
ldi     ZH,high(2*TEXT_E)
rcall   Kirim_Perintah
rcall   Baris2
rcall   KirimPesan_LCD
rcall   delay1d
rcall   delay1d

ldi     tmp,DISPCLR
rcall   Kirim_Perintah
RCALL   DELAY1d
ret

;*********************************************************
;Subrutin inisialisasi usart
;*********************************************************
init_usart:
ldi     tmp,high(ubbr_value)          ;set baud rate
out     UBRRH,tmp
ldi     tmp,low(ubbr_value)
out     UBRRL,tmp
ldi     tmp,(1<<RXEN)|(1<<TXEN);|(1<<RXCIE)          ;enable receiver and
transmitter
out     UCSRB,tmp
ldi     tmp,(1<<URSEL)|(3<<UCSZ0)       ;set frame format: 8data, 1 stop bit
out     UCSRC,tmp
ret

;*********************************************************
;usart transmit data
;*********************************************************
usart_tx:
sbis    UCSRA,UDRE          ;wait for empty transmit buffer
rjmp    usart_tx
out     UDR,txbyte    ;Put data into buffer, sends the data
ret

;*********************************************************
;usart receive data
;*********************************************************
usart_rx:
sbis    UCSRA,RXC ;wait for data to be received
rjmp    usart_rx
```

```
in      rxbyte,UDR    ;get and return received data from buffer
ret

;*********************************************************
;Subrutin kirim pesan ke LCD
;*********************************************************
KirimPesan_LCD:
clr     tmp

LoopKirimPesan_LCD:
lpm
mov     tmp,r0
cpi     tmp,0
brne    Kirim_LCD
ret

Kirim_LCD:
rcall   Kirim_Karakter
adiw    ZL,1
rjmp    LoopKirimPesan_LCD

Kirim_Perintah:
cbi             portc,RS_LCD
rcall   Kirim_DataLCD
rcall   delay5
ret

Kirim_Karakter:
cbi     portc,RW_LCD
sbi     portc,RS_LCD
rcall   Kirim_DataLCD
rcall   delay5
ret

Kirim_DataLCD:
sbi     portc,E_LCD
out     porta,tmp
cbi     portc,E_LCD
clr     tmp
ret

;*********************************************************
;LCD pindah baris 2
;*********************************************************
Baris2:
ldi     tmp,0xC0
```

```
rcall   Kirim_Perintah
ret

;*********************************************************
;LCD pindah baris 1
;*********************************************************
Baris1:
ldi     tmp,0x80
rcall   Kirim_Perintah
ret

;*********************************************************
;Subrutin inisialisasi LCD
;*********************************************************
Init_LCD1:
     ldi        tmp,DISPCLR
     rcall      Kirim_Perintah
     RCALL  DELAY5

     ldi        tmp,FUNCSET
     rcall      Kirim_Perintah
     RCALL  DELAY5

     ldi        tmp,DISPON
     rcall      Kirim_Perintah
     RCALL  DELAY5

     ldi        tmp,ENTRMOD
     rcall      Kirim_Perintah
     RCALL  DELAY5

     ldi        tmp,DISPCLR
     rcall      Kirim_Perintah
     RCALL  DELAY5
ret

;*********************************************************
;Subrutin delay 5ms
;*********************************************************
delay5:
ldi     r16,0b00000100              ;aktifkan enable interupt
out     TIMSK,r16
ldi     r16,high(timer_value) ;masukkan nilai timer
out     TCNT1H,r16
ldi     r16,LOW(timer_value)
out     TCNT1L,r16
```

```
ldi     r16,0b00000101              ;masukkan prescaler utk timer disini 1024
out     TCCR1B,R16


looptimer:
in      r17,TIFR
sbrs    r17,TOV1
rjmp    looptimer
ldi     r16,0b00000100
out     TIFR,r16
ret

;*********************************************************
;Subrutin delay 100US
;*********************************************************
delay100u:
ldi     r16,0b00000100                  ;aktifkan enable interupt
out     TIMSK,r16
ldi     r16,high(timer_value100)    ;masukkan nilai timer
out     TCNT1H,r16
ldi     r16,LOW(timer_value100)
out     TCNT1L,r16
ldi     r16,0b00000001                  ;masukkan prescaler utk timer disini
1024
out     TCCR1B,R16

looptimer3:
in      R17,TIFR
sbrs    r17,TOV1            ;tunggu sampai timer1 overflow flag set
rjmp    looptimer3
ldi     r16,0b00000100              ;timer 1 overflow flag dinolkan dg beri
logik1
out     TIFR,r16
ret

;*********************************************************
;Subrutin delay 1d
;*********************************************************
delay1d:
ldi     r16,0b00000100
out     TIMSK,r16
ldi     r16,high(timer_value1)
out     TCNT1H,r16
ldi     r16,low(timer_value1)
out     TCNT1L,r16
ldi     r16,0b00000101
```

```
out     TCCR1B,r16

looptimer2:
in      r17,TIFR
sbrs    r17,TOV1
rjmp    looptimer2
ldi     r16,0b00000100
out     TIFR,r16
ret

TEXT_1:     .db     "  PROGRAMMABLE",0
TEXT_2:     .db     "         TIMER",0

TEXT_3:     .db     "     PERALATAN",0
TEXT_4:     .db     "       LISTRIK",0

TEXT_5:     .db     "     JARAK JAUH",0
TEXT_6:     .db     "       VIA SMS",0

TEXT_7:     .db     "    Ranu Setyobudi ",0
TEXT_8:     .db     "     NIM: 0017062 ",0

TEXT_9:     .db     "  Dosen Pembimbing",0
TEXT_A:     .db     "    Ir. F Yudi L, MT",0

TEXT_B:     .db     "     T. ELKA S-1 ",0
TEXT_C:     .db     "    ITN MALANG",0

TEXT_D:     .db     "       SKRIPSI",0
TEXT_E:     .db     "         2009",0

;*********************************************************
reset_all:
;*********************************************************
ldi     tmp,0
mov     P1,tmp
mov     P2,tmp
mov     T1,tmp
mov     T2,tmp
mov     H1,tmp
mov     M1,tmp
mov     S1,tmp
mov     H2,tmp
mov     M2,tmp
mov     S2,tmp
mov     A1,tmp
```

```
mov   U7S,tmp
mov   BCDH,tmp
mov   BCDL,tmp
mov   txbyte,tmp
mov   rxbyte,tmp
ret

;*********************************************************
reset_T1:
;*********************************************************
ldi   tmp,0
mov   P1,tmp
mov   T1,tmp
mov   H1,tmp
mov   M1,tmp
mov   S1,tmp
mov   U7S,tmp
ret

;*********************************************************
reset_T2:
;*********************************************************
ldi   tmp,0
mov   P2,tmp
mov   T2,tmp
mov   H2,tmp
mov   M2,tmp
mov   S2,tmp
mov   U7S,tmp
ret

;*********************************************************
usart_rxc:
;*********************************************************
CBI   PORTB,3
rcall usart_rx
mov   tmp,rxbyte
rcall kirim_karakter
CBI   PORTB,3
push  r16
in          r16,sreg
push  r16
loop_usart:
rjmp  tes2
rcall usart_rx
cpi   rxbyte,$31
```

```
brne    loop_usart
cbi     portb,3
rcall   cek_sms
pop     r16
out     sreg,r16
pop     r16
reti
seting:
ldi     r21,0b10000000
out     UCSRA,r21
sei
rjmp    mulai

kirim_csms:
ldi     zl,low(2*msg_csms)
ldi     zh,high(2*msg_csms)

csms:
lpm
mov     txbyte,r0
cpi     txbyte,0
breq    tunggus
rcall   usart_tx

;rcall   usart_rx ;>>>test loopback

adiw zh:zl,1
rjmp    csms

msg_csms:
.db     "AT+CSMS=1",13,10,0        ;(13+jml karakter)

tunggu:
rcall   usart_rx
cpi     rxbyte,'K'
brne    tunggus
rcall   usart_rx
rcall   usart_rx

;*********************************************************
kirim_cnmi:
;*********************************************************
ldi     zl,low(2*msg_cnmi)
ldi     zh,high(2*msg_cnmi)

cnmi:
```

```
lpm
mov    txbyte,r0
cpi    txbyte,0
breq   tungguss
rcall  usart_tx

;rcall  usart_rx ;>>>test loopback

adiw zh:zl,1
rjmp   cnmi

msg_cnmi:
.db    "AT+CNMI=1,1,0,1,1",13,10,0        ;(13+jml karakter)

tungguss:
rcall  usart_rx
cpi    rxbyte,'K'
brne   tungguss
rcall  usart_rx
rcall  usart_rx
ret
```

INDEKS

DAFTAR PUSTAKA

1. Atmel. 2006. ATMega 8535 AVR. USA: www.atmel.com
2. Atmel. 2006. AVR and Third Party Tools. USA: www.atmel.com
3. Atmel. 2006. AVR Assembler User Guide. USA: www.atmel.com
4. Atmel. 2006. AVR Hardware Design Considerations. USA: www.atmel.com
5. Atmel. 2006. AVR Instruction Set. USA: www.atmel.com
6. Eko, Agfianto., 2002, Belajar Mikrokontroller AT89C51/52/55 Teori dan Aplikasi, Gava Media, Yogyakarta
7. Siemens AG. 2001. AT Command Set for Siemens Mobile Phones and Modems. Munich: www.siemens.com
8. Wardana, Lingga., 2004, Perancangan Sistem Alarm Kendaraan dan Rumah Berbasis *Short Message Service* (SMS), Media Elektro, Yogyakarta

LAMPIRAN

JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S1

KONSENTRASI	T. ELEKTRONIKA

1.	Nama Mahasiswa	RANU SETYOBUDI		NIM	0017062
2.	Keterangan Pelaksanaan	Tanggal: 24-10-2008	Waktu:		Tempat / Ruang:

Spesifikasi Judul (berilah tanda silang) *)

3.	a. Sistem Tenaga Elektrik	e. Embbeded System		i. Sistem Informasi	
	b. Konversi Energi	f. Antar Muka		j. Jaringan Komputer	
	c. Sistem Kendali	g. Elektronika Telekomunikasi		k. Web	
	d. Tegangan Tinggi	h. Elektronika Instrumentasi		l. Algoritma Cerdas	

4.	Judul Proposal yang diseminarkan Mahasiswa	PROGRAMMABLE TIMER PERALATAN LISTRIK YANG DIAKSES VIA SMS
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	.....
6.	Catatan :	2. TAMBAH RTC
	Catatan :	.....

Persetujuan Judul Skripsi

Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II
.....	.....

Mengetahui,
Ketua Jurusan

Ir. F. Yudi Limpraptono,MT
NIP. Y. 1039500274

Disetujui,
Calon Dosen Pembimbing

Pembimbing I	Pembimbing II
.....	.....

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

ERO) MALANG
GA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 27 Oktober 2008

Nomor : **ITN-422 /7/TA /2008**
Lampiran :
Perihal : ***Bimbingan Skripsi***

Kepada : Yth. Sdr. **IR. F. YUDI LIMPRAPTONO, MT**
Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skipsi untuk mahasiswa:

Nama	: **RANU SETYOBUDI**
Nim	: **0017062**
Fakultas	: **Teknologi Industri**
Jurusan	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Elektronika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu 6 (enam) bulan, terhitung mulai tanggal:

24 OKTOBER 2008 S/D 24 APRIL 2009

Sebagai satu syarat untuk menempuh Ujian sarjana.
Demikian atas perhatian serta kerjasama yang baik kami ucapkan terima kasih

Ketua Jurusan
Teknik Elektro S-1

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
ITN
JURUSAN TEKNIK ELEKTRO
STRATA - 1

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

Tindasan:
1. Mahasiswa yang Bersangkutan
2. Arsip

Form S-4a

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANGLO KM. 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : RANU SETYOBUDI
NIM : 0017062
Masa Bimbingan : 24 Oktober 2008 s/d 24 April 2009
Judul Skripsi : PROGRAMMABLE TIMER PERALATAN LISTRIK YANG DIAKSES VIA SMS

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	2/3 '09	Bab I — V Revisi	
2		Bab II	
3		Bab III	
4		Bab IV	
5		Bab V	
6		Seminar	
7		Acc Alhir	
8			
9			
10			

Malang,
Dosen Pembimbing

Ir. F. Yudi Limpraptono, MT
NIP.Y. 1039 5900274

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Reno setyobudi
NIM : 0017062
Perbaikan meliputi :

- Abstrak bhs Inggris hilangkan

Malang, 200

(_____)

tures

h-performance, Low-power AVR® 8-bit Microcontroller
anced RISC Architecture
130 Powerful Instructions – Most Single Clock Cycle Execution
32 x 8 General Purpose Working Registers
Fully Static Operation
Up to 16 MIPS Throughput at 16 MHz
On-chip 2-cycle Multiplier
volatile Program and Data Memories
8K Bytes of In-System Self-Programmable 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
512 Bytes EEPROM
  Endurance: 100,000 Write/Erase Cycles
512 Bytes Internal SRAM
Programming Lock for Software Security
pheral Features
Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
Real Time Counter with Separate Oscillator
Four PWM Channels
8-channel, 10-bit ADC
  8 Single-ended Channels
  7 Differential Channels for TQFP Package Only
  2 Differential Channels with Programmable Gain at 1x, 10x, or 200x for TQFP Package Only
Byte-oriented Two-wire Serial Interface
Programmable Serial USART
Master/Slave SPI Serial Interface
Programmable Watchdog Timer with Separate On-chip Oscillator
On-chip Analog Comparator
ial 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
nd Packages
32 Programmable I/O Lines
40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad MLF
ating Voltages
2.7 - 5.5V for ATmega8535L
4.5 - 5.5V for ATmega8535
d Grades
0 - 8 MHz for ATmega8535L
0 - 16 MHz for ATmega8535

8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

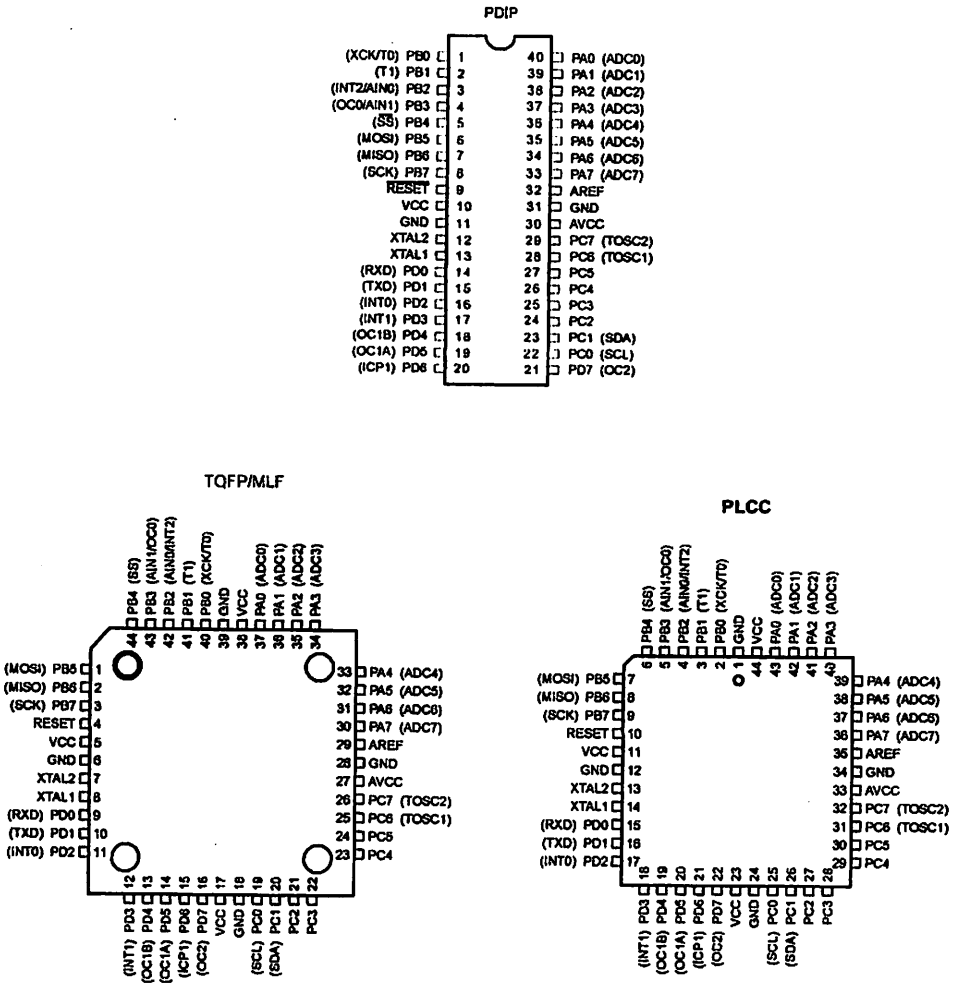
ATmega8535
ATmega8535L

Advance Information

Rev. 2502C–AVR–04/03

Configurations

Figure 1. Pinout ATmega8535



imer

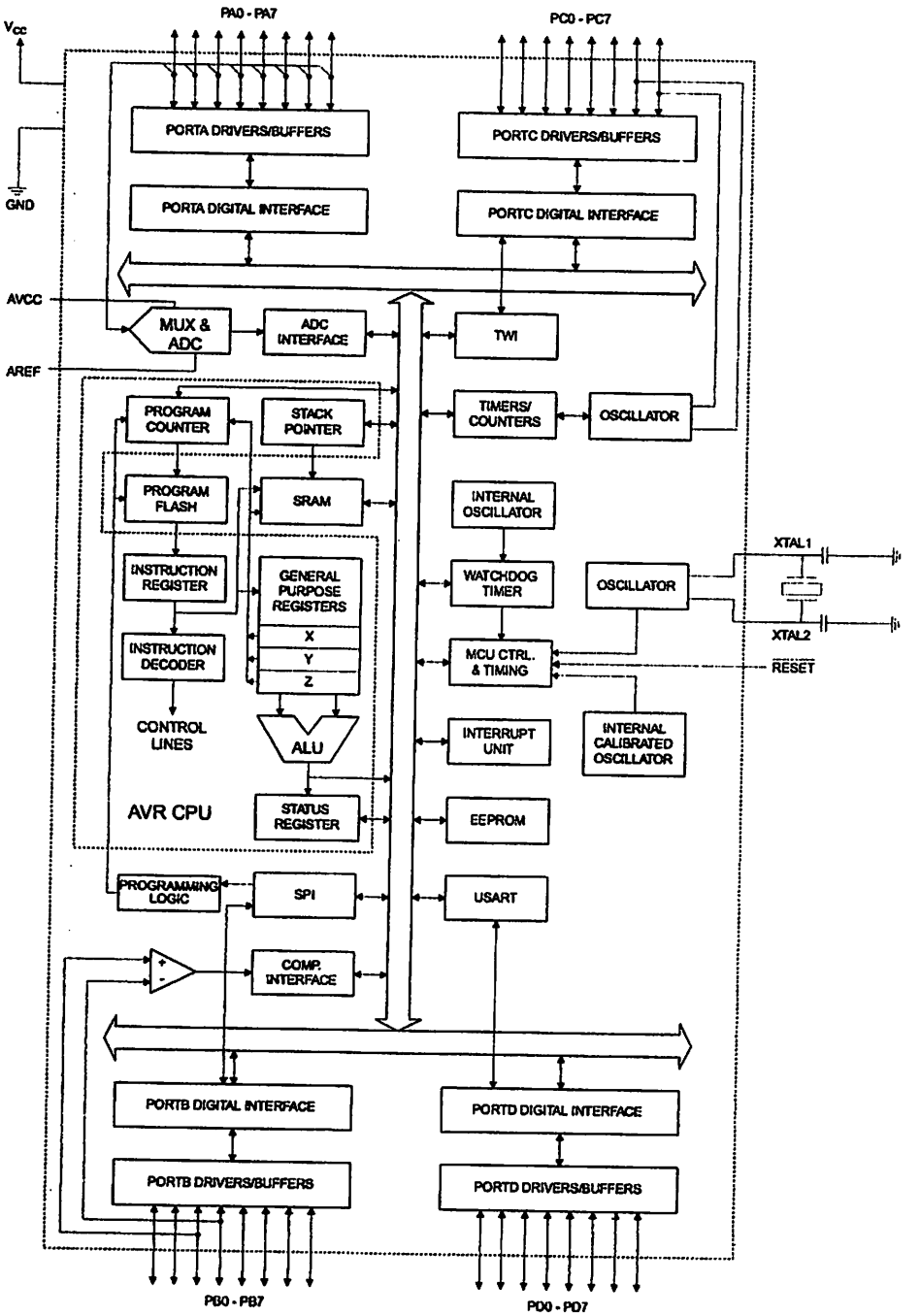
Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

rview

The ATmega8535 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing instructions in a single clock cycle, the ATmega8535 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

k Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8535 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain in TQFP package, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8535 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega8535 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

S8535 Compatibility

The ATmega8535 provides all the features of the AT90S8535. In addition, several new features are added. The ATmega8535 is backward compatible with AT90S8535 in most cases. However, some incompatibilities between the two microcontrollers exist. To solve this problem, an AT90S8535 compatibility mode can be selected by programming the S8535C fuse. ATmega8535 is pin compatible with AT90S8535, and can replace the AT90S8535 on current Printed Circuit Boards. However, the location of fuse bits and the electrical characteristics differs between the two devices.

8535 Compatibility

Programming the S8535C fuse will change the following functionality:

- The timed sequence for changing the Watchdog Time-out period is disabled. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 43 for details.
- The double buffering of the USART Receive Register is disabled. See "AVR USART vs. AVR UART – Compatibility" on page 142 for details.

Descriptions

	Digital supply voltage.
	Ground.
A (PA7..PA0)	Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.
B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega8535 as listed on page 57.
C (PC7..PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.
D (PD7..PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega8535 as listed on page 61.
RST	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 35. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.

›ut Code mples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.

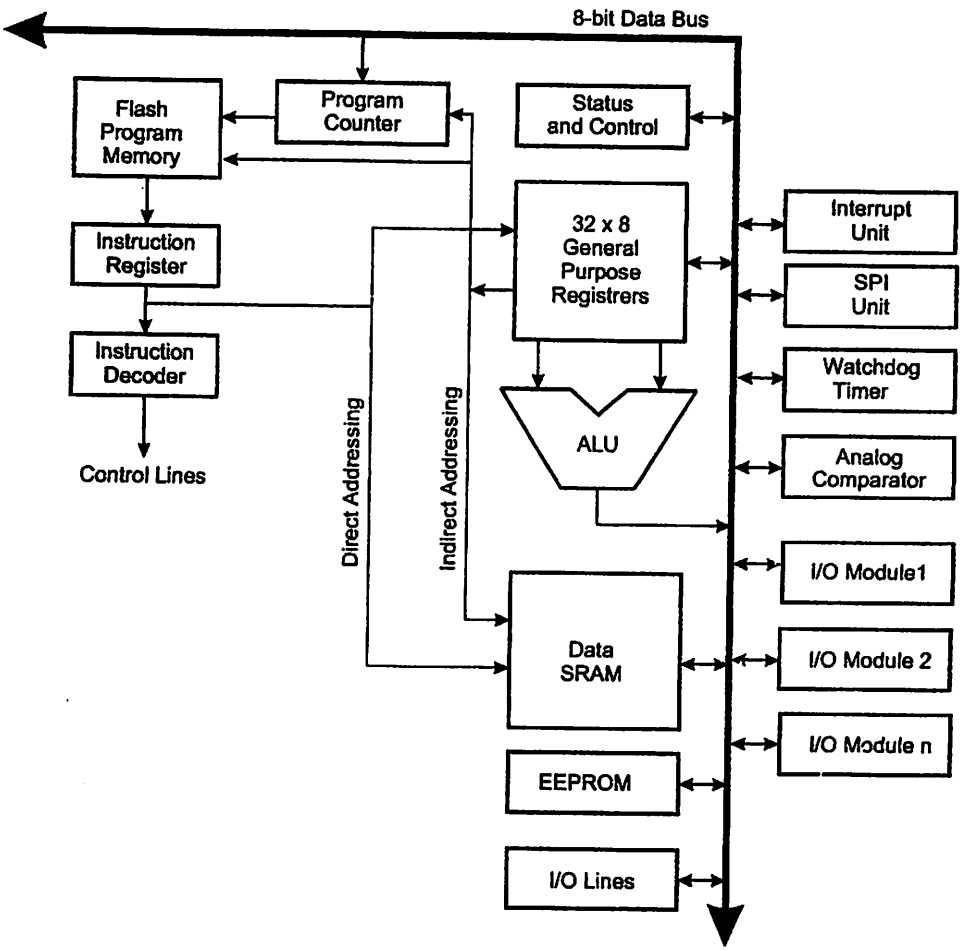
‹ CPU Core

›duction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

itectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept

enables instructions to be executed in every clock cycle. The program memory is In-System Re-Programmable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-registers, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F.

- Arithmetic Logic

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

us Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will, in many cases, remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half carry is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

- **Bit 4 – S: Sign Bit, $S = N \oplus V$**

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

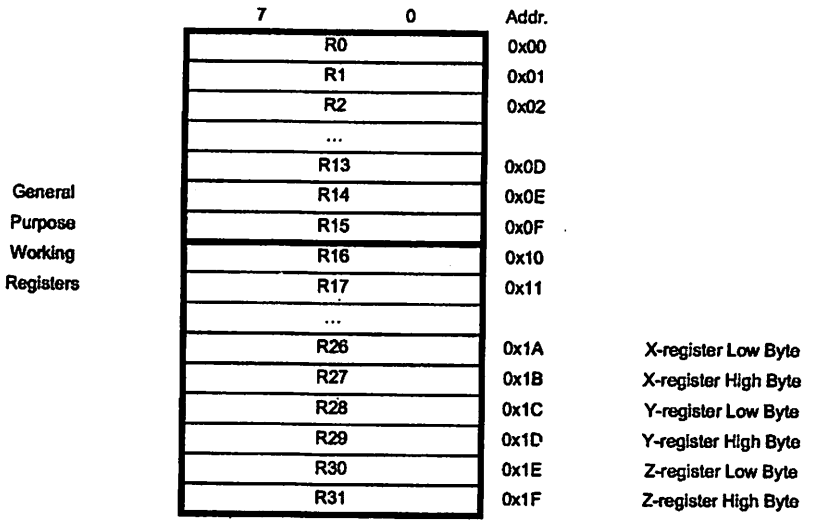
ɜral Purpose ster File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers



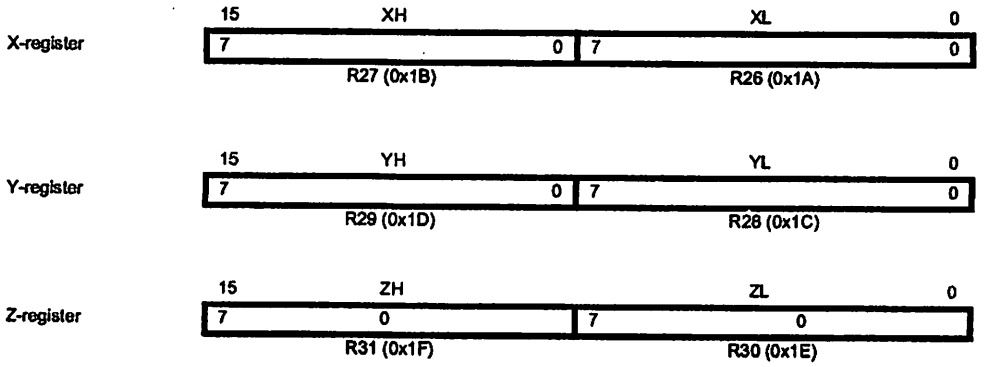
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

-register, Y-register, and ister

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-, Y-, and Z-registers



In the different addressing modes, these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	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	

uction Execution ng

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

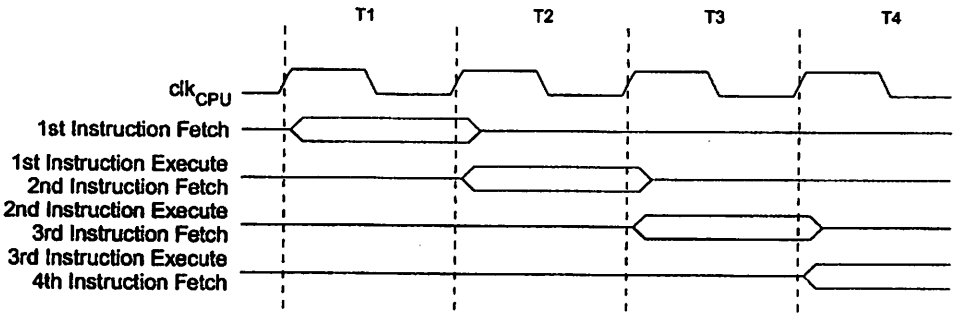
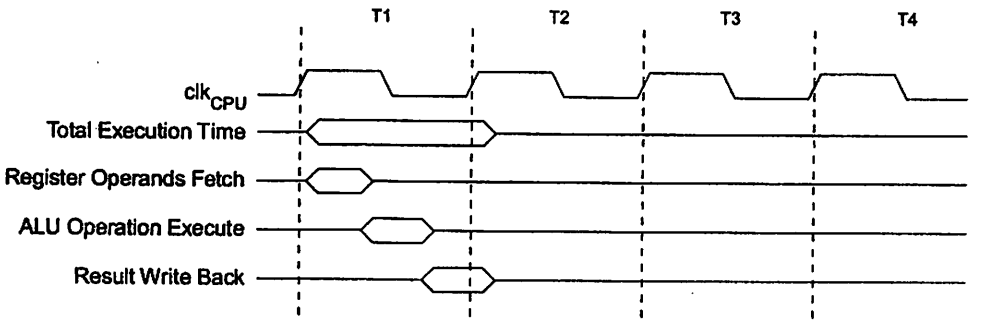


Figure 7 shows the internal timing concept for the Register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



t and Interrupt ling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate Program Vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 233 for details.

The lowest addresses in the program memory space are, by default, defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 44. The list also determines the priority levels of the different interrupts. The lower the address, the higher the priority level is. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to "Interrupts" on page 44 for more information. The Reset Vector can

also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 220.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in   r16, SREG      ; store SREG value
cli                 ; disable interrupts during timed sequence
sbi  EECR, EEMWE    ; start EEPROM write
sbi  EECR, EEWE
out  SREG, r16      ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;  /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example
sei *; set global interrupt enable* **sleep** *; enter sleep, waiting for interrupt* ; note: will enter sleep before any pending ; interrupt(s)
C Code Example
_SEI(); */* set global interrupt enable */* _SLEEP(); */* enter sleep, waiting for interrupt */* /* note: will enter sleep before any pending interrupt(s) */

upt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles, the Program Vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The Vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

ATmega8535 nories

This section describes the different memories in the ATmega8535. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega8535 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

stem ogrammable Flash ram Memory

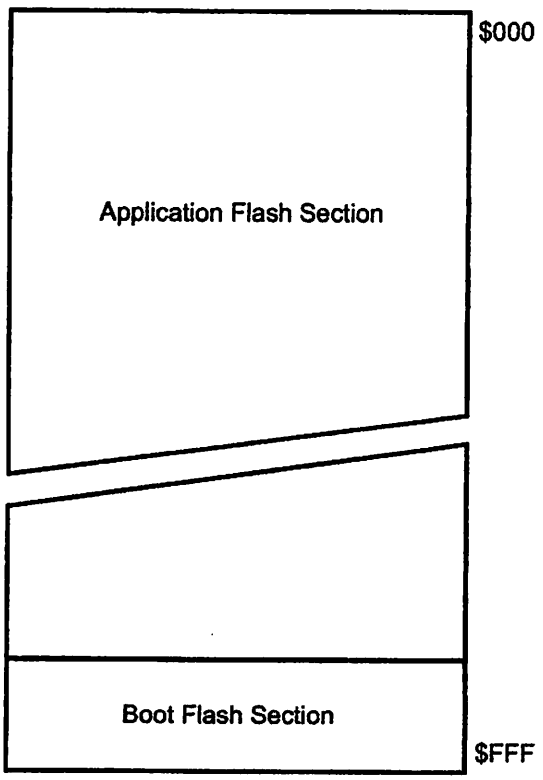
The ATmega8535 contains 8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8535 Program Counter (PC) is 12 bits wide, thus addressing the 4K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 220. "Memory Programming" on page 233 contains a detailed description on Flash Programming in SPI or Parallel Programming mode.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 11.

Figure 8. Program Memory Map



M Data Memory

Figure 9 shows how the ATmega8535 SRAM Memory is organized.

The 608 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 512 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

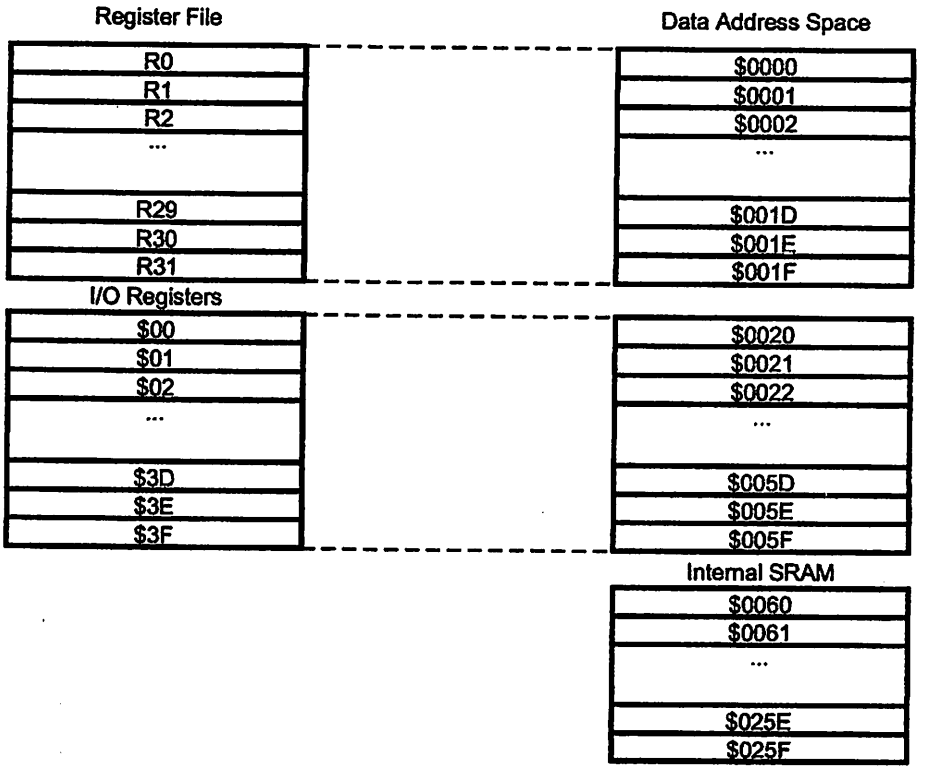
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 512 bytes of internal data SRAM in the ATmega8535 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 9.

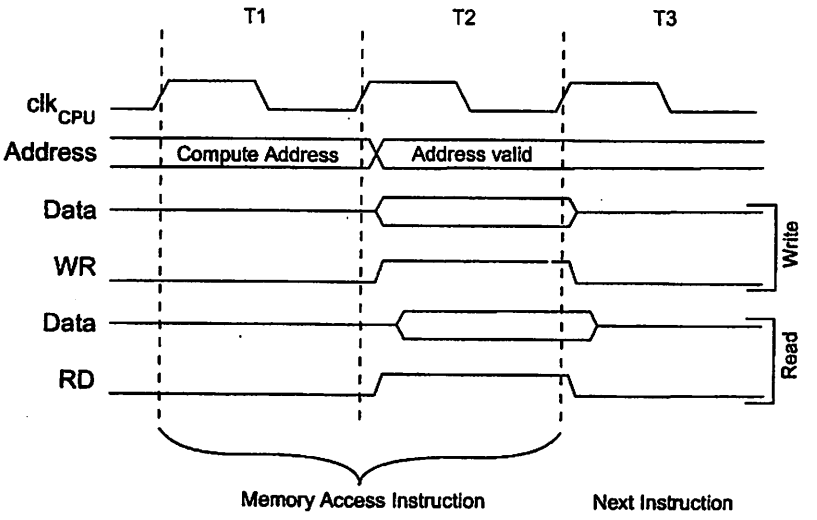
Figure 9. Data Memory Map



Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega8535 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

"Memory Programming" on page 233 contains a detailed description on EEPROM Programming in SPI or Parallel Programming mode.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device, for some period of time, to run at a voltage lower than specified as minimum for the clock frequency used, see "Preventing EEPROM Corruption" on page 20 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

:EPROM Address ;ter – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	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	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8535 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

EPROM Data Register –

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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	

- **Bits 7..0 – EEDR7..0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

:PROM Control Register R

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8535 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I-bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never updated by the CPU, step 2 can be omitted. See "Boot Loader Support – Read-While-Write Self-Programming" on page 220 for details about Boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles[1]	Typ Programming Time
EEPROM Write (from CPU)	8448	8.4 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse settings.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to Data Register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up Address and Data Registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in Address Register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from Data Register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up Address Register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from Data Register */
  return EEDR;
}
```

M Write During Power-leep Mode

When entering Power-down sleep mode while an EEPROM write operation is active, the EEPROM write operation will continue, and will complete before the write access time has passed. However, when the write operation is completed, the Oscillator continues running, and as a consequence, the device does not enter Power-down entirely. It is therefore recommended to verify that the EEPROM write operation is completed before entering Power-down.

ting EEPROM tion

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

lemory

The I/O space definition of the ATmega8535 is shown in page 260.

All ATmega8535 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

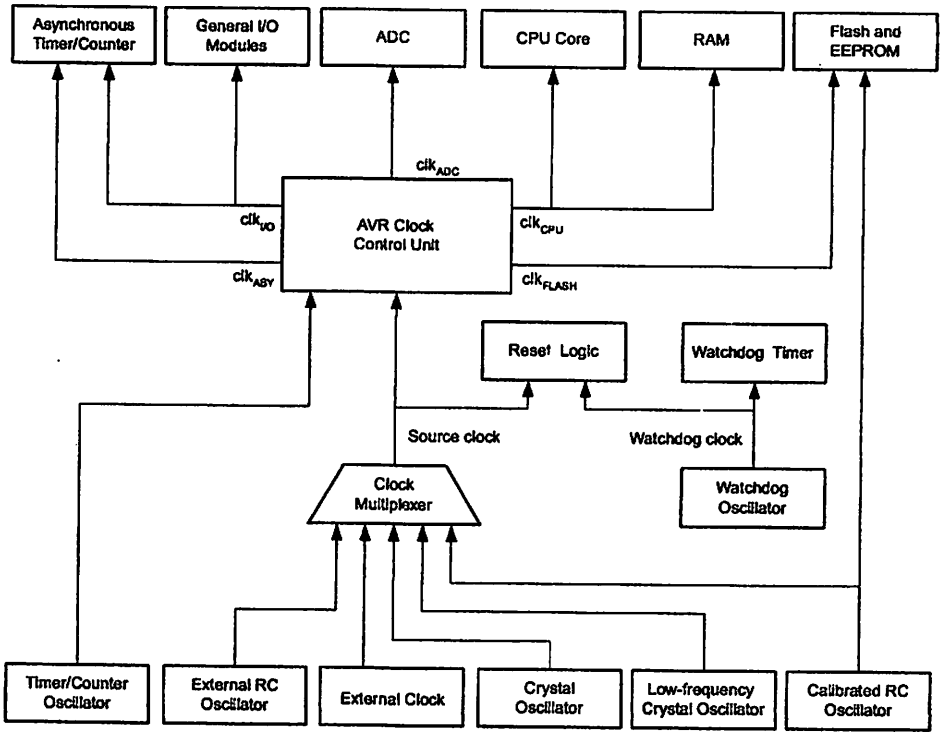
The I/O and peripherals control registers are explained in later sections.

tem Clock and
:k Options

k Systems and their
ibution

Figure 11 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 30. The clock systems are detailed below.

Figure 11. Clock Distribution



lock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

ck – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when $clk_{I/O}$ is halted, enabling TWI address reception in all sleep modes.

lock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

chronous Timer Clock –

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

k Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 2. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 3. The frequency of the Watchdog Oscillator is voltage dependent as shown in "ATmega8535 Typical Characteristics – Preliminary Data" on page 259.

Table 3. Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

ılt Clock Source

The device is shipped with CKSEL = "0001" and SUT = "10". The default clock source setting is therefore the Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programmer.

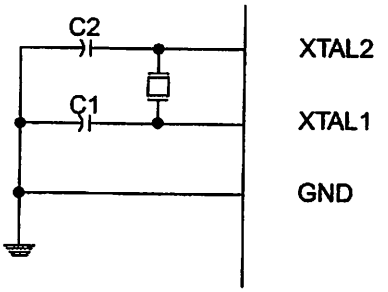
:al Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 12. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably.

This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 4. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 12. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 4.

Table 4. Crystal Oscillator Operating Modes

CKOPT	CKSEL3..1	Frequency Range[1] (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[2]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 - 16.0	12 - 22

Notes: 1. The frequency ranges are preliminary values.
2. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 5.

Table 5. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK[2]	–	Ceramic resonator, BOD enabled
0	11	1K CK[2]	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK[2]	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.

2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

-frequency Crystal llator

To use a 32.768 kHz watch crystal as the clock source for the device, the Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to "1001". The crystal should be connected as shown in Figure 12. By programming the CKOPT Fuse, the user can enable internal capacitors on XTAL1 and XTAL2, thereby removing the need for external capacitors. The internal capacitors have a nominal value of 36 pF.

When this Oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 6.

Table 6. Start-up Times for the Low-frequency Crystal Oscillator Clock Selection

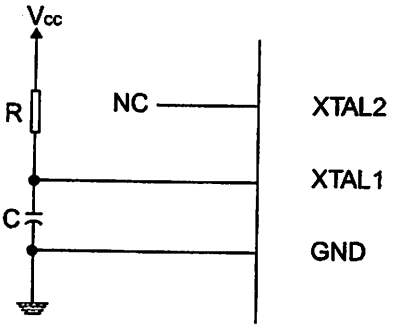
SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	1K CK(1)	4.1 ms	Fast rising power or BOD enabled
01	1K CK(1)	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11	Reserved		

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

nal RC Oscillator

For timing insensitive applications, the external RC configuration shown in Figure 13 can be used. The frequency is roughly estimated by the equation f = 1/(3RC). C should be at least 22 pF. By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND, thereby removing the need for an external capacitor. For more information on Oscillator operation and details on how to choose R and C, refer to the External RC Oscillator application note.

Figure 13. External RC Configuration



The Oscillator can operate in four different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..0 as shown in Table 7.

Table 7. External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	- 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

When this Oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 8.

Table 8. Start-up Times for the External RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	18 CK	–	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK(1)	4.1 ms	Fast rising power or BOD enabled

Note: 1. This option should not be used when operating close to the maximum frequency of the device.

rated Internal RC lator

The calibrated internal RC Oscillator provides a fixed 1.0, 2.0, 4.0, or 8.0 MHz clock. All frequencies are nominal values at 5V and 25°C. This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 9. If selected, it will operate with no external components. The CKOPT Fuse should always be unprogrammed when using this clock option. During Reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 5V, 25°C and 1.0 MHz Oscillator frequency selected, this calibration gives a frequency within ± 1% of the nominal frequency. When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 235.

Table 9. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

Note: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 10. XTAL1 and XTAL2 should be left unconnected (NC).

Table 10. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10[1]	6 CK	65 ms	Slowly rising power
11	Reserved		

Note: 1. The device is shipped with this option selected.

ıtor Calibration Register
CAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7..0 – CAL7..0: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. During Reset, the 1 MHz calibration value which is located in the signature row high byte (address 0x00) is automatically loaded into the OSCCAL Register. If the internal RC is used at other frequencies, the calibration values must be loaded manually. This can be done by first reading the signature row by a programmer, and then store the calibration values in the Flash or EEPROM. Then the value can be read by software and loaded into the OSCCAL Register.

When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing 0xFF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0, 2.0, 4.0, or 8.0 MHz. Tuning to other values is not guaranteed, as indicated in Table 11.

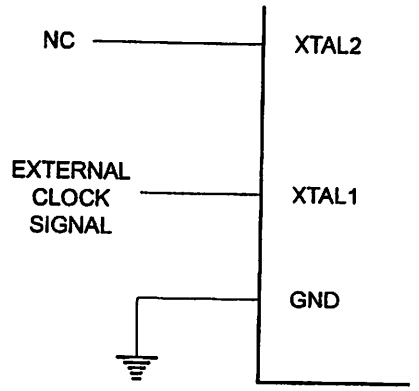
Table 11. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)
0x00	50	100
0x7F	75	150
0xFF	100	200

rnal Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 14. To run the device on an external clock, the CKSEL Fuses must be programmed to "0000". By programming the CKOPT Fuse, the user can enable an internal 36 pF capacitor between XTAL1 and GND.

Figure 14. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 12.

Table 12. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11	Reserved		

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behaviour. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

'Counter Oscillator

For AVR microcontrollers with Timer/Counter Oscillator pins (TOSC1 and TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock source to TOSC1 is not recommended.

er Management Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the six sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the MCUCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be activated by the SLEEP instruction. See Table 13 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, it executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a Reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 11 on page 22 presents the different clock systems in the ATmega8535, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

ontrol Register – R

The MCU Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bits 7, 5, 4 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the six available sleep modes as shown in Table 13.

Table 13. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby mode and Extended Standby mode are only available with external crystals or resonators.

- **Bit 6 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

lode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, USART, Analog Comparator, ADC, Two-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ADC bit in the Analog Comparator Control and Status register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

Noise Reduction

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or INT1, or an external interrupt on INT2 can wake up the MCU from ADC Noise Reduction mode.

r-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the External Oscillator is stopped, while the External Interrupts, the Two-wire Serial Interface address watch, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, an external level interrupt on INT0 or INT1, or an external interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to "External Interrupts" on page 65 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL fuses that define the Reset Time-out period, as described in "Clock Sources" on page 23.

-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is clocked asynchronously, i.e., the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the Global Interrupt Enable bit in SREG is set.

If the asynchronous timer is NOT clocked asynchronously, Power-down mode is recommended instead of Power-save mode because the contents of the registers in the

asynchronous timer should be considered undefined after wake-up in Power-save mode if AS2 is 0.

This sleep mode basically halts all clocks except clk_{ASY}, allowing operation only of asynchronous modules, including Timer/Counter2 if clocked asynchronously.

dby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

nded Standby Mode

When the SM2..0 bits are 111 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles.

14. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock domains					Oscillators		Wake up sources					
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT2 INT1 INT0	TWI Address Match	Timer 2	SPM/ EEPROM Ready	A D C	Other I/O
			X	X	X	X	X(2)	X	X	X	X	X	X
ction				X	X	X	X(2)	X(3)	X	X	X	X	
r-								X(3)	X				
r-					X(2)		X(2)	X(3)	X	X(2)			
by(1)						X		X(3)	X				
ded by(1)					X(2)	X	X(2)	X(3)	X	X(2)			

1. External Crystal or resonator selected as clock source
2. If AS2 bit in ASSR is set
3. Only INT2 or level interrupt INT1 and INT0

nizing Power sumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

g-to-Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "Analog-to-Digital Converter" on page 202 for details on ADC operation.

g Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 199 for details on how to configure the Analog Comparator.

-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 37 for details on how to configure the Brown-out Detector.

al Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 39 for details on the start-up time.

dog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 39 for details on how to configure the Watchdog Timer.

ns

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section "Digital Input Enable and Sleep Modes" on page 53 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

em Control and
#t

ing the AVR

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be an RJMP instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 15 shows the reset logic. Table 15 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 23.

Sources

The ATmega8535 has four sources of Reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.

Figure 15. Reset Logic

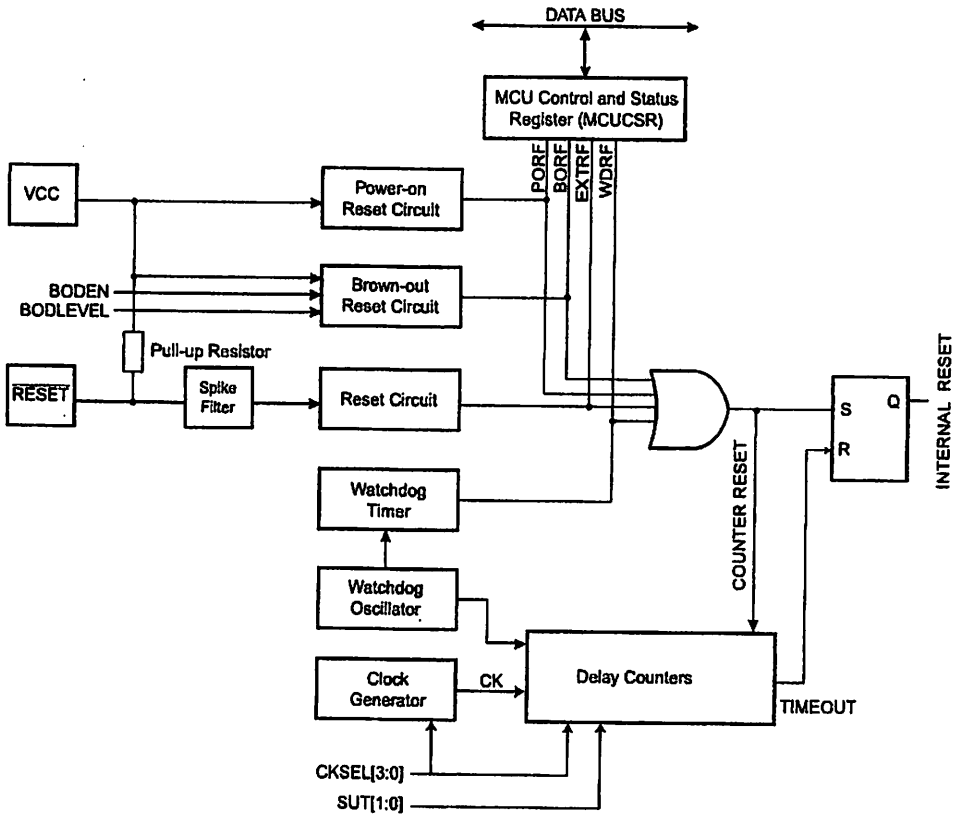


Table 15. Reset Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			1.4	2.3	V
	Power-on Reset Threshold Voltage (falling)[2]			1.3	2.3	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.1		0.9	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin			50		ns
V_{BOT}	Brown-out Reset Threshold Voltage[3]	BODLEVEL = 1	2.5	2.7	3.2	V
		BODLEVEL = 0	3.7	4.0	4.2	
t_{BOD}	Minimum low voltage period for Brown-out Detection	BODLEVEL = 1		2		µs
		BODLEVEL = 0		2		µs
V_{HYST}	Brown-out Detector hysteresis			130		mV

Notes:
1. Values are guidelines only.
2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).
3. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the

production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 1 for ATmega8535L and BODLEVEL = 0 for ATmega8535. BODLEVEL = 1 is not applicable for ATmega8535.

r-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 16. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

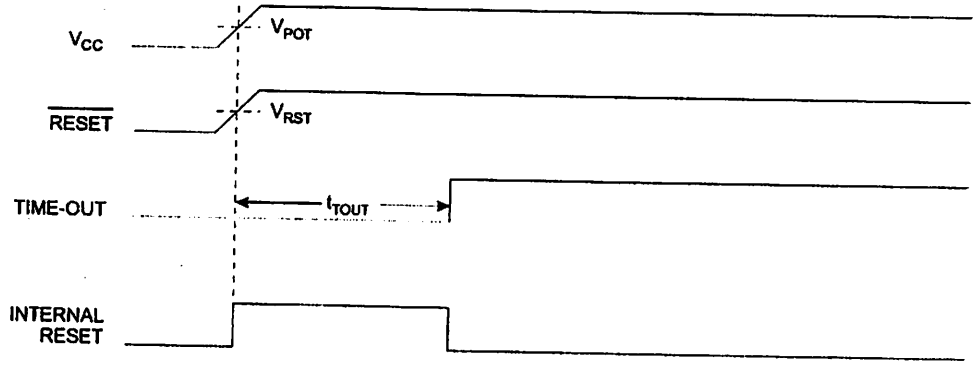
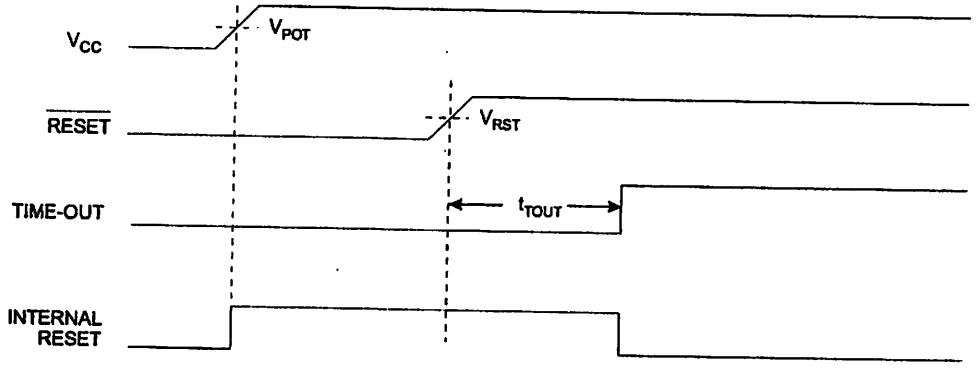


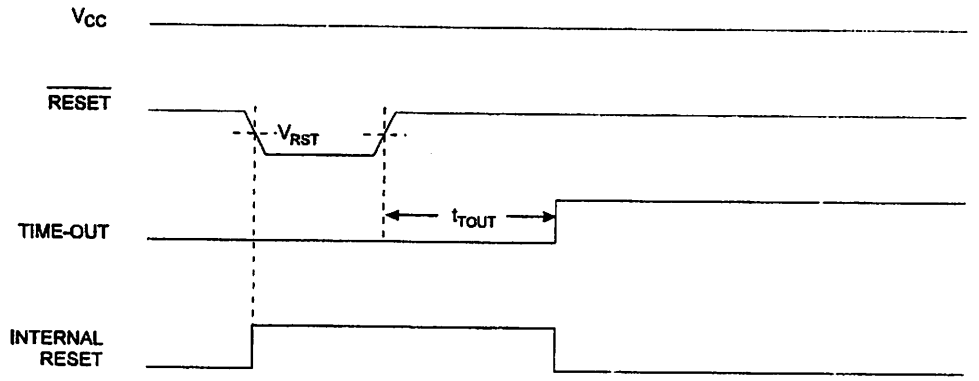
Figure 17. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



al Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} on its positive edge, the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 18. External Reset During Operation



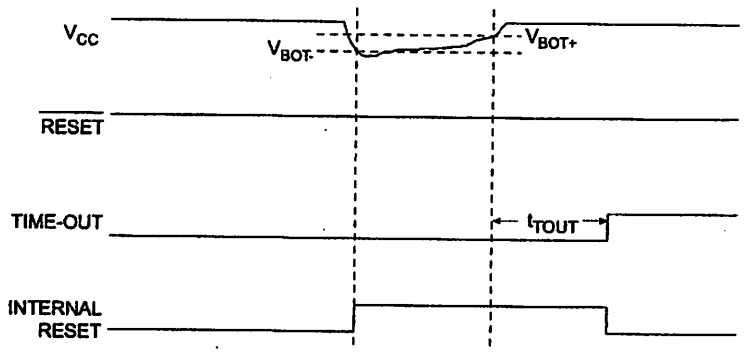
out Detection

ATmega8535 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 19), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 19), the delay counter starts the MCU after the time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 15.

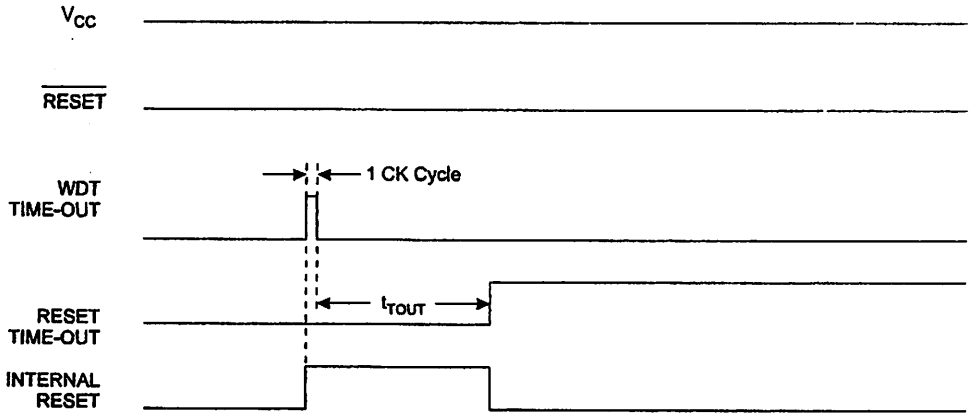
Figure 19. Brown-out Reset During Operation



dog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 39 for details on operation of the Watchdog Timer.

Figure 20. Watchdog Reset During Operation



Control and Status er – MCUCSR

The MCU Control and Status Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0		See Bit Description				

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

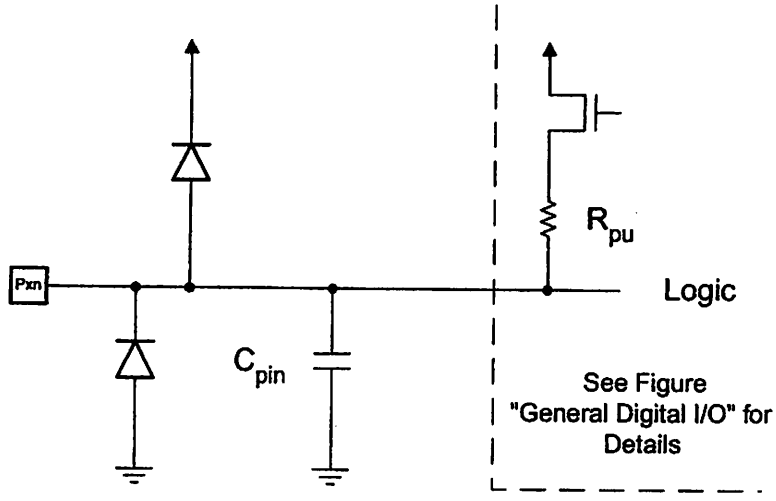
To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUCSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

orts

luction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 22. Refer to "Electrical Characteristics" on page 251 for a complete list of parameters.

Figure 22. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description for I/O-Ports" on page 63.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

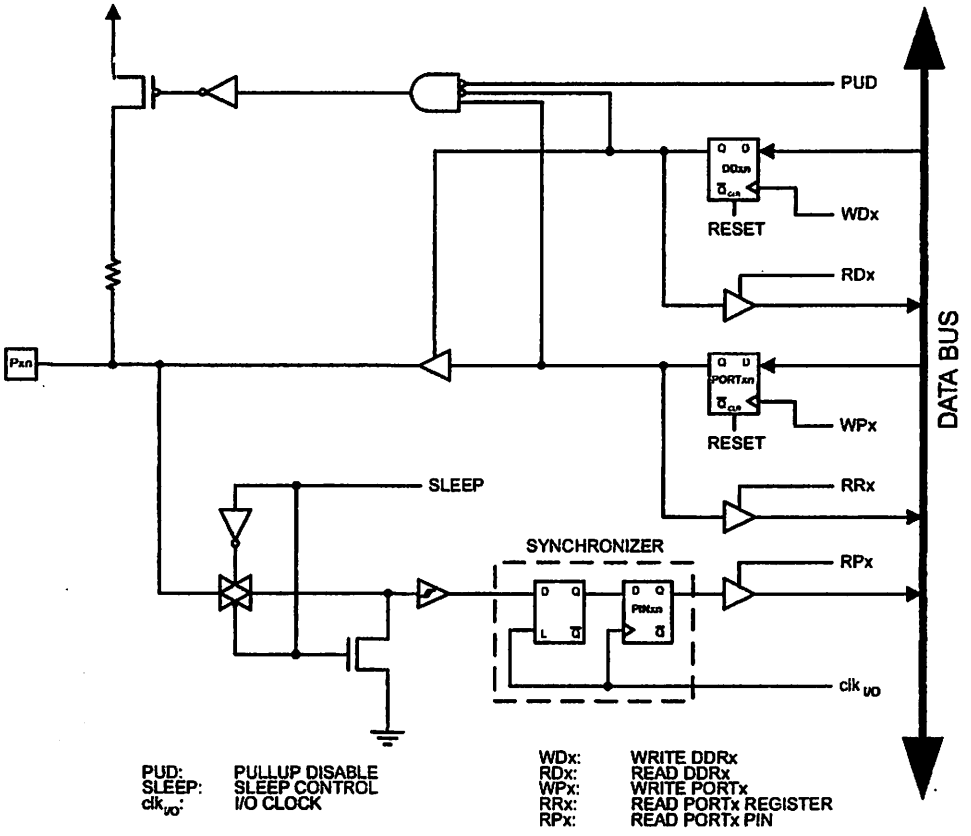
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 50. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 54. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

as General Digital

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 23 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 23. General Digital I/O[1]



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

uring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O-Ports" on page 63, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written a logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written a logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written a logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} =

0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 21 summarizes the control signals for the pin value.

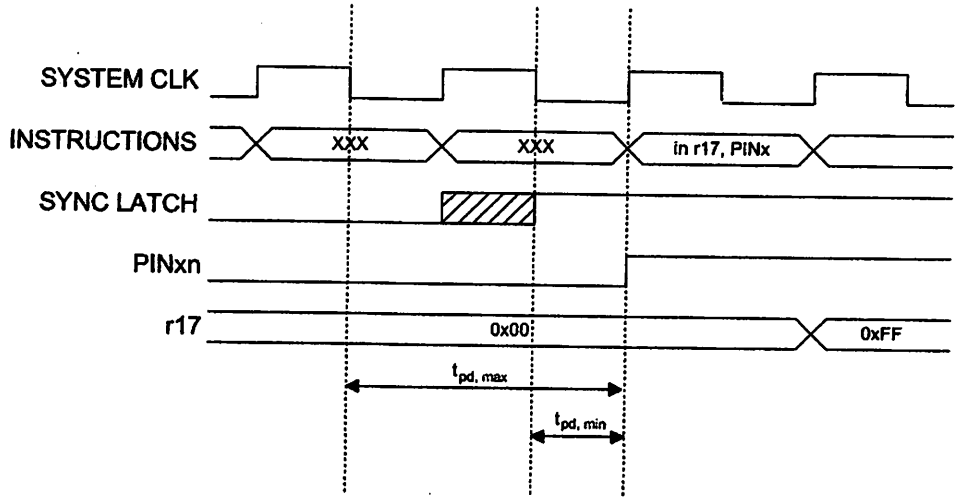
Table 21. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

g the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 23, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 24 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 24. Synchronization when Reading an Externally Applied Pin Value

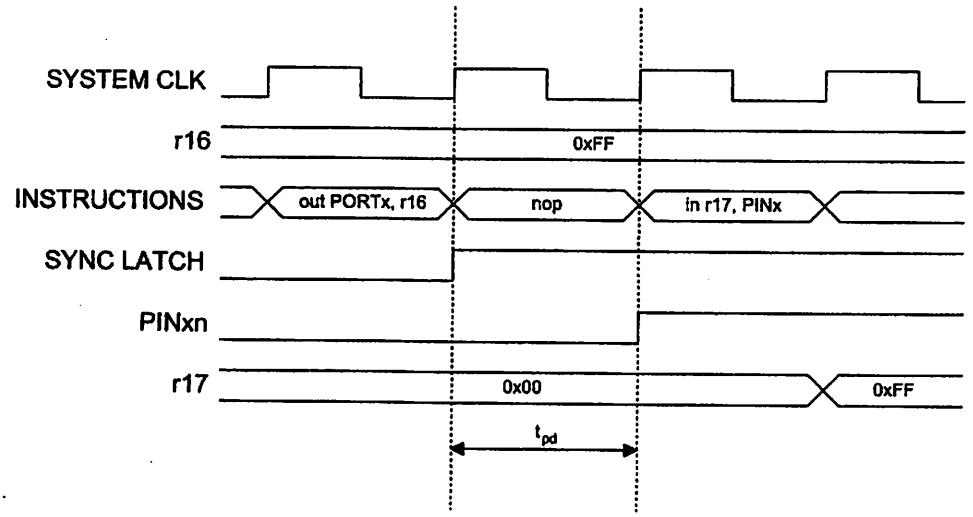


Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the suc-

ceeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a *nop* instruction must be inserted as indicated in Figure 25. The *out* instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is one system clock period.

Figure 25. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a *nop* instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi     r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi     r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out     PORTB,r16
out     DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in      r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bits 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

nput Enable and Sleep

As shown in Figure 23, the digital input signal can be clamped to ground at the input of the schmitt trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in Power-down mode, Power-save mode, Standby mode, and Extended Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as External Interrupt pins. If the External Interrupt Request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 54.

If a logic high level ("one") is present on an Asynchronous External Interrupt pin configured as "Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin" while the external interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.

ınected pins

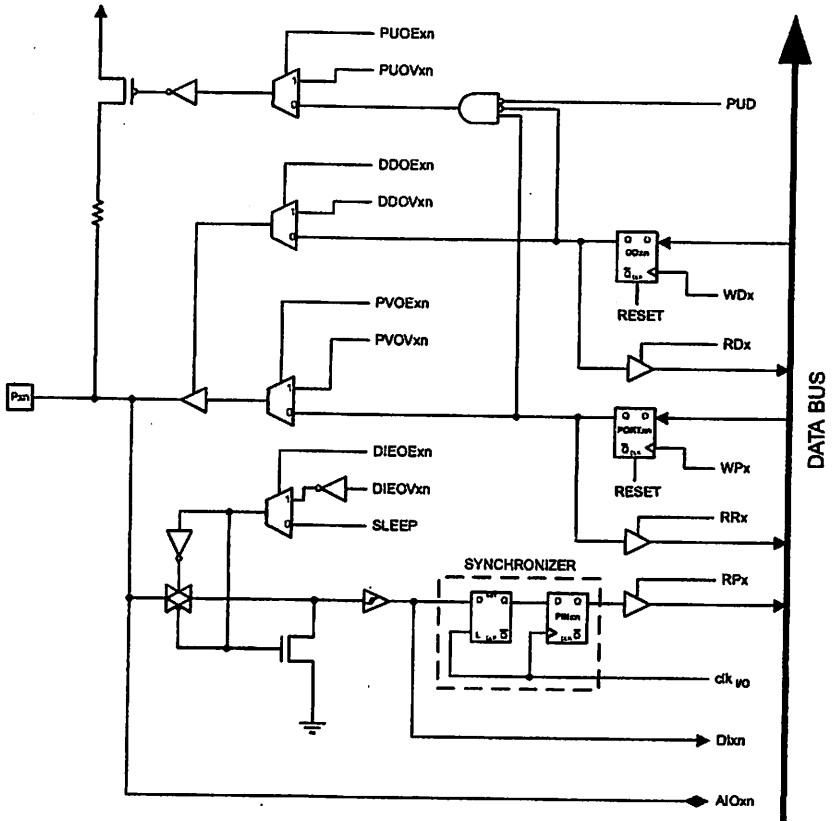
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pullup. In this case, the pullup will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pullup or pulldown. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

ıate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 26 shows how the port pin control signals from the simplified Figure 23 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 26. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE
SLEEP:	SLEEP CONTROL

PUD:	PULLUP DISABLE
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WPx:	WRITE PORTx
RPx:	READ PORTx PIN
$clk_{I/O}$:	I/O CLOCK
Dixn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 22 summarizes the function of the overriding signals. The pin and port indexes from Figure 26 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 22. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU-state (Normal mode, sleep modes).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep modes).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/output	This is the Analog Input/Output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

al Function IO Register –

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2 – PUD: Pull-up disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See "Configuring the Pin" on page 50 for more details about this feature.

ate Functions of Port A

Port A has an alternate function as analog input for the ADC as shown in Table 23. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

Table 23. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

Table 24 and Table 25 relate the alternate functions of Port A to the overriding signals shown in Figure 26 on page 54.

Table 24. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/ADC7	PA6/ADC6	PA5/ADC5	PA4/ADC4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT

Table 25. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/ADC3	PA2/ADC2	PA1/ADC1	PA0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

ate Functions Of Port B

The Port B pins with alternate functions are shown in Table 26.

Table 26. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

The alternate pin configuration is as follows:

- **SCK – Port B, Bit 7**

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB7. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB7 bit.

- **MISO – Port B, Bit 6**

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB6. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB6 bit.

- **MOSI – Port B, Bit 5**

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

- **$\overline{SS}$ – Port B, Bit 4**

$\overline{SS}$: Slave Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit.

- **AIN1/OC0 – Port B, Bit 3**

AIN1, Analog Comparator Negative input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

OC0, Output Compare Match output: The PB3 pin can serve as an external output for the Timer/Counter0 Compare Match. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC0 pin is also the output pin for the PWM mode timer function.

- **AIN0/INT2 – Port B, Bit 2**

AIN0, Analog Comparator Positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

INT2, External Interrupt Source 2: The PB2 pin can serve as an external interrupt source to the MCU.

- **T1 – Port B, Bit 1**

T1, Timer/Counter1 Counter Source.

- **T0/XCK – Port B, Bit 0**

T0, Timer/Counter0 Counter Source.

XCK, USART External Clock. The Data Direction Register (DDB0) controls whether the clock is output (DDB0 set) or input (DDB0 cleared). The XCK pin is active only when the USART operates in synchronous mode.

Table 27 and Table 28 relate the alternate functions of Port B to the overriding signals shown in Figure 26 on page 54. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 27. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/SCK	PB6/MISO	PB5/MOSI	PB4/$\overline{SS}$
PUOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
PUOV	PORTB7 • $\overline{PUD}$	PORTB6 • $\overline{PUD}$	PORTB5 • $\overline{PUD}$	PORTB4 • $\overline{PUD}$
DDOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
DDOV	0	0	0	0
PVOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR INPUT	SPI SLAVE INPUT	SPI $\overline{SS}$
AIO	–	–	–	–

Table 28. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC0/AIN1	PB2/INT2/AIN0	PB1/T1	PB0/T0/XCK
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0 ENABLE	0	0	UMSEL
PVOV	OC0	0	0	XCK OUTPUT
DIEOE	0	INT2 ENABLE	0	0
DIEOV	0	1	0	0
DI	–	INT2 INPUT	T1 INPUT	XCK INPUT/ T0 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

te Functions of Port C

The Port C pins with alternate functions are shown in Table 29.

Table 29. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

The alternate pin configuration is as follows:

- **TOSC2 – Port C, Bit 7**

TOSC2, Timer Oscillator pin 2: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC7 is disconnected from the port, and becomes the inverting output of the Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- **TOSC1 – Port C, Bit 6**

TOSC1, Timer Oscillator pin 1: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC6 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- **SDA – Port C, Bit 1**

SDA, Two-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC1 bit.

- **SCL – Port C, Bit 0**

SCL, Two-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the Two-wire Serial Interface, pin PC0 is disconnected from the port and becomes the Serial Clock I/O pin for the Two-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation. When this pin is used by the Two-wire Serial Interface, the pull-up can still be controlled by the PORTC0 bit.

Table 30 and Table 31 relate the alternate functions of Port C to the overriding signals shown in Figure 26 on page 54.

Table 30. Overriding Signals for Alternate Functions in PC7..PC6

Signal Name	PC7/TOSC2	PC6/TOSC1
PUOE	AS2	AS2
PUOV	0	0
DDOE	AS2	AS2
DDOV	0	0
PVOE	0	0
PVOV	0	0
DIEOE	AS2	AS2
DIEOV	0	0
DI	–	–
AIO	T/C2 OSC OUTPUT	T/C2 OSC INPUT

Table 31. Overriding Signals for Alternate Functions in PC1..PC0[1]

Signal Name	PC1/SDA	PC0/SCL
PUOE	TWEN	TWEN
PUOV	PORTC1 • $\overline{PUD}$	PORTC0 • $\overline{PUD}$
DDOE	TWEN	TWEN
DDOV	SDA_OUT	SCL_OUT
PVOE	TWEN	TWEN
PVOV	0	0
DIEOE	0	0
DIEOV	0	0
DI	–	–
AIO	SDA INPUT	SCL INPUT

Note: 1. When enabled, the Two-wire Serial Interface enables slew-rate controls on the output pins PC0 and PC1. This is not shown in the figure. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

ıte Functions of Port D

The Port D pins with alternate functions are shown in Table 32.

Table 32. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

The alternate pin configuration is as follows:

- **OC2 – Port D, Bit 7**

OC2, Timer/Counter2 Output Compare Match output: The PD7 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDD7 set (one)) to serve this function. The OC2 pin is also the output pin for the PWM mode timer function.

- **ICP1 – Port D, Bit 6**

ICP1 – Input Capture Pin: The PD6 pin can act as an Input Capture pin for Timer/Counter1.

- **OC1A – Port D, Bit 5**

OC1A, Output Compare Match A output: The PD5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **OC1B – Port D, Bit 4**

OC1B, Output Compare Match B output: The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt Source 1: The PD3 pin can serve as an external interrupt source.

- **INT0 – Port D, Bit 2**

INT0, External Interrupt Source 0: The PD2 pin can serve as an external interrupt source.

- **TXD – Port D, Bit 1**

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

- **RXD – Port D, Bit 0**

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

Table 33 and Table 34 relate the alternate functions of Port D to the overriding signals shown in Figure 26 on page 54.

Table 33. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/OC2	PD6/ICP1	PD5/OC1A	PD4/OC1B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2 ENABLE	0	OC1A ENABLE	OC1B ENABLE
PVOV	OC2	0	OC1A	OC1B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	ICP1 INPUT	–	–
AIO	–	–	–	–

Table 34. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 • $\overline{PUD}$
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	–	RXD
AIO	–	–	–	–

ster Description for orts

Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Data Direction Register A

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

Input Pins Address –

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

Data Direction Register B

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

Input Pins Address –

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
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	

Data Direction Register C

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
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	

Input Pins Address –

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

Data Direction Register)

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

nput Pins Address –

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

ıory
ɟramming

ram And Data
ory Lock Bits

The ATmega8535 provides six Lock bits which can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 97. The Lock bits can only be erased to "1" with the Chip Erase command.

Table 96. Lock Bit Byte[1]

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 97. Lock Bit Protection Modes[2]

Memory Lock Bits			Protection Type
LB Mode	**LB2**	**LB1**	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.[1]
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.[1]
BLB0 Mode	**BLB02**	**BLB01**	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If interrupt vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
BLB1 Mode	**BLB12**	**BLB11**	

Table 97. Lock Bit Protection Modes[2] (Continued)

Memory Lock Bits			Protection Type
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Notes: 1. Program the Fuse bits before programming the Lock bits.
2. "1" means unprogrammed, "0" means programmed.

Bits

The ATmega8535 has two Fuse bytes. Table 98 and Table 99 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 98. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
S8535C	7	Select AT90S8535 compatibility mode	1 (unprogrammed)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN[1]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 93 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 93 for details)	0 (programmed)[3]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in Serial Programming mode.
2. The CKOPT Fuse functionality depends on the setting of the CKSEL bits. See "Clock Sources" on page 23. for details.
3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 93 on page 231.

Table 99. Fuse Low Byte

Fuse Low Byte	Bit no	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

Notes: 1. The default value of SUT1..0 results in maximum start-up time. See Table 10 on page 28 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 1 MHz. See Table 2 on page 23 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit 1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

ıg of Fuses

The Fuse values are latched when the device enters Programming mode and changes of the Fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

ture Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both Serial and Parallel mode, also when the device is locked. The three bytes reside in a separate address space.

For the ATmega8535 the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel)
2. 0x001: 0x93 (indicates 8 KB Flash memory)
3. 0x002: 0x08 (indicates ATmega8535 device when 0x001 is 0x93)

ation Byte

The ATmega8535 stores four different calibration values for the internal RC Oscillator. These bytes resides in the signature row high byte of the addresses 0x000, 0x0001, 0x0002, and 0x0003 for 1, 2, 4, and 8 MHz respectively. During Reset, the 1 MHz value is automatically loaded into the OSCCAL Register. If other frequencies are used, the calibration value has to be loaded manually, see "Oscillator Calibration Register – OSCCAL" on page 28 for details.

el Programming
ıeters, Pin
ing, and
ıands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega8535. Pulses are assumed to be at least 250 ns unless otherwise noted.

Names

In this section, some pins of the ATmega8535 are referenced by signal names describing their functionality during parallel programming, see Figure 115 and Table 100. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 102.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 103.

Figure 115. Parallel Programming

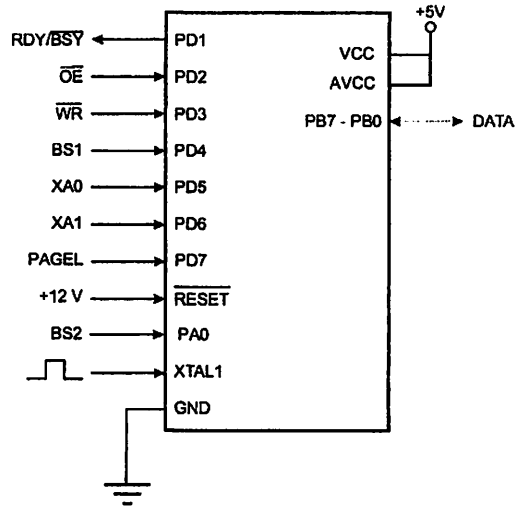


Table 100. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active low)
$\overline{WR}$	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 ("0" selects low byte, "1" selects high byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program Memory and EEPROM data Page Load
BS2	PA0	I	Byte Select 2 ("0" selects low byte, "1" selects 2'nd high byte)
DATA	PB7 - 0	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low)

Table 101. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 102. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1)
0	1	Load Data (High or Low data byte for Flash determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Table 103. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse Bits
0010 0000	Write Lock Bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes and Calibration byte
0000 0100	Read Fuse and Lock Bits
0000 0010	Read Flash
0000 0011	Read EEPROM

Table 104. No. of Words in a Page and no. of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 105. No. of Words in a Page and no. of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

lel Programming

Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND, and wait at least 100 µs.
2. Set $\overline{\text{RESET}}$ to "0" and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 101 on page 237 to "0000" and wait at least 100 ns.
4. Apply 11.5 - 12.5V to $\overline{\text{RESET}}$. Any activity on Prog_enable pins within 100 ns after +12V has been applied to $\overline{\text{RESET}}$, will cause the device to fail entering Programming mode.

Note, if External Crystal or External RC configuration is selected, it may not be possible to apply qualified XTAL1 pulses. In such cases, the following algorithm should be followed:

1. Set Prog_enable pins listed in Table 101 on page 237 to "0000".
2. Apply 4.5 - 5.5V between V_{CC} and GND simultanously as 11.5 - 12.5V is applied to $\overline{\text{RESET}}$.
3. Wait 100 ns.
4. Re-program the fuses to ensure that External Clock is selected as clock source (CKSEL3:0 = 0b0000) If Lock bits are programmed, a Chip Erase command must be executed before changing the fuses.
5. Exit Programming mode by power the device down or by bringing $\overline{\text{RESET}}$ pin to 0b0.
6. Entering Programming mode with the original algorithm, as described above.

erations for Efficient nming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

ase

The Chip Erase will erase the Flash and EEPROM[1] memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or the EEPROM is reprogrammed.

Note: 1. The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command "Chip Erase"

1. Set XA1, XA0 to "10". This enables command loading.
2. Set BS1 to "0".
3. Set DATA to "1000 0000". This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{\text{WR}}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{\text{BSY}}$ goes low.
6. Wait until RDY/$\overline{\text{BSY}}$ goes high before loading a new command.

mming the Flash

The Flash is organized in pages, see Table 104 on page 237. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command "Write Flash"

1. Set XA1, XA0 to "10". This enables command loading.
2. Set BS1 to "0".
3. Set DATA to "0001 0000". This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B. Load Address Low byte

1. Set XA1, XA0 to "00". This enables address loading.
2. Set BS1 to "0". This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to "01". This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to "1". This selects high data byte.
2. Set XA1, XA0 to "01". This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to "1". This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (See Figure 117 for signal waveforms.)

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 116 on page 240. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a page write.

G. Load Address High byte

1. Set XA1, XA0 to "00". This enables address loading.
2. Set BS1 to "1". This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

1. Set BS1 to "0".
2. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$goes low.

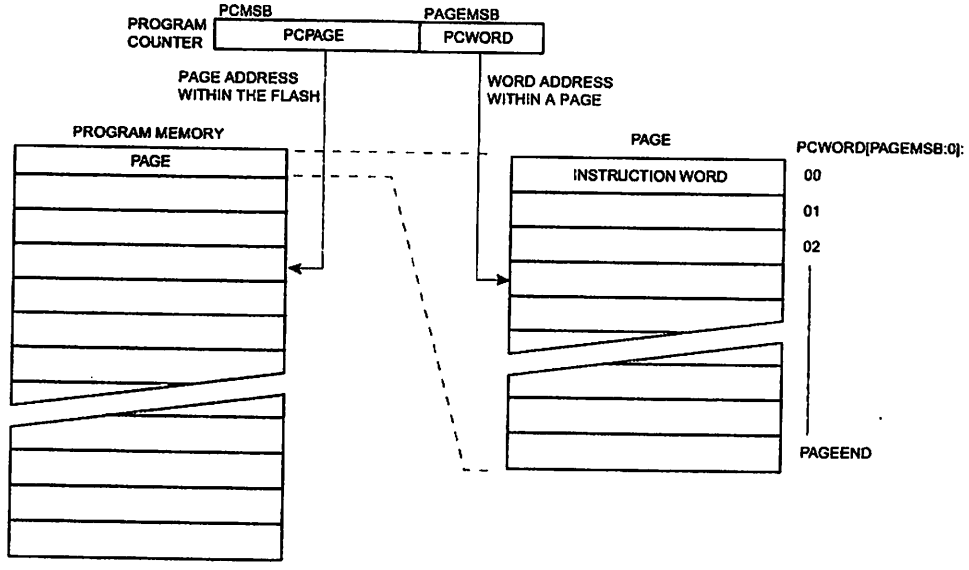
3. Wait until RDY/$\overline{BSY}$ goes high. (See Figure 117 for signal waveforms)

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

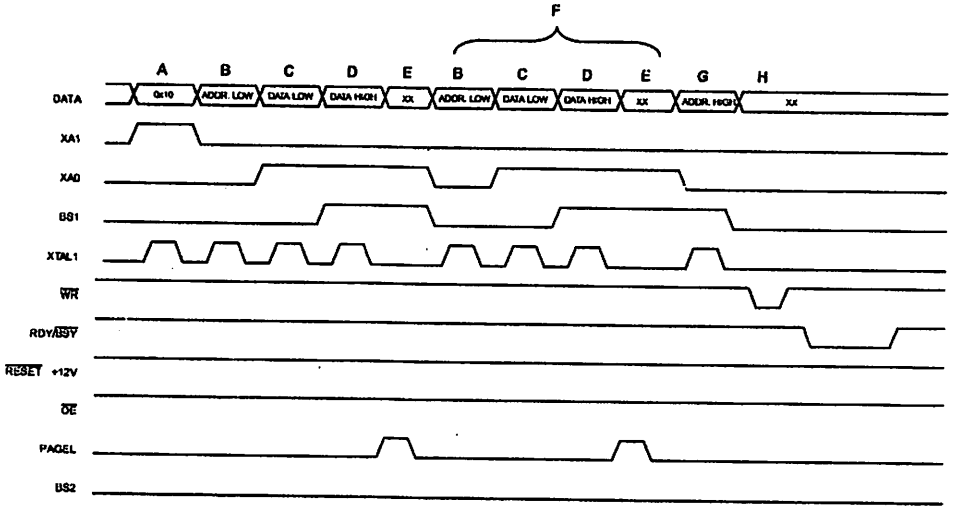
1. 1. Set XA1, XA0 to "10". This enables command loading.
2. Set DATA to "0000 0000". This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 116. Addressing the Flash which is Organized in Pages[1]



Note: 1. PCPAGE and PCWORD are listed in Table 104 on page 237.

Figure 117. Programming the Flash Waveforms[1]



Note: 1. "XX" is don't care. The letters refer to the programming description above.

mming the EEPROM

The EEPROM is organized in pages, see Table 105 on page 237. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 239 for details on Command, Address and Data loading):

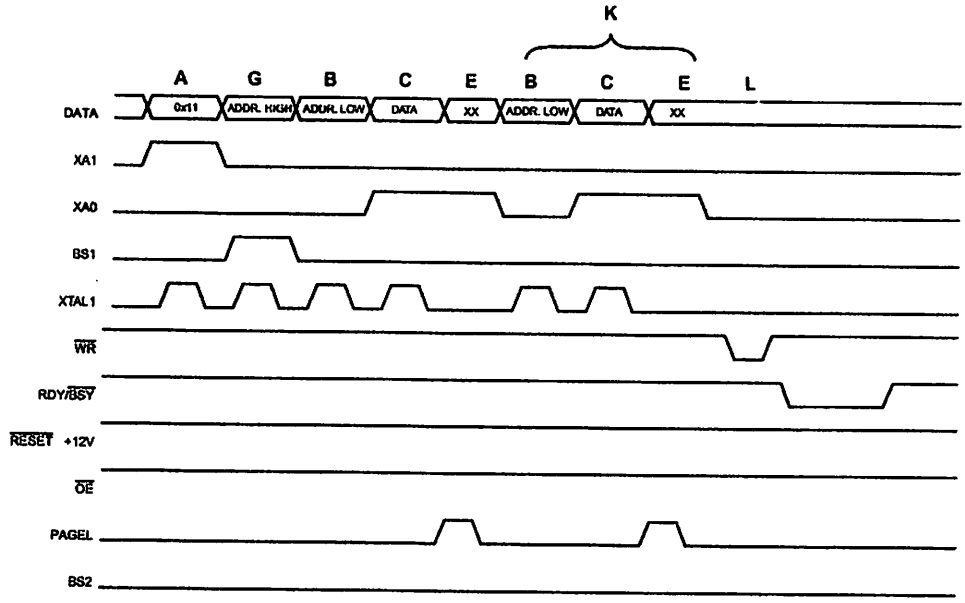
1. A: Load Command "0001 0001".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page.

1. Set BS1 to "0".
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page. (See Figure 118 for signal waveforms.)

Figure 118. Programming the EEPROM Waveforms



l the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. A: Load Command "0000 0010".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to "0", and BS1 to "0". The Flash word low byte can now be read at DATA.
5. Set BS1 to "1". The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to "1".

ng the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. A: Load Command "0000 0011".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to "0", and BS1 to "0". The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to "1".

mming the Fuse Low

The algorithm for programming the Fuse Low bits is as follows (refer to "Programming the Flash" on page 239 for details on Command and Data loading):

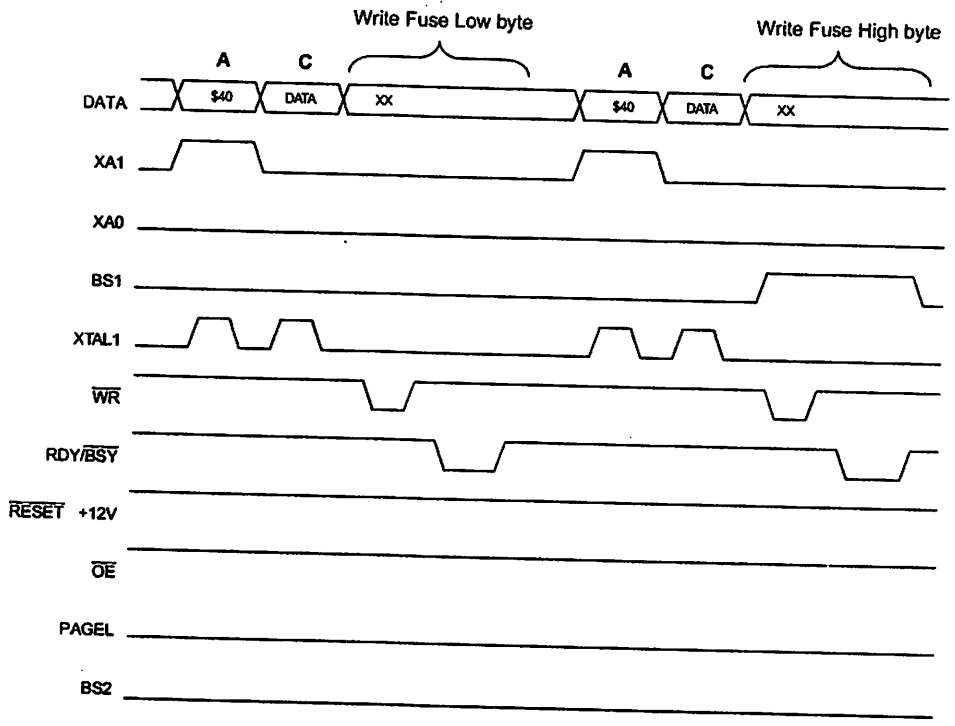
1. A: Load Command "0100 0000".
2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Set BS1 to "0" and BS2 to "0". This selects low data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

nming the Fuse High

The algorithm for programming the Fuse high bits is as follows (refer to "Programming the Flash" on page 239 for details on Command and Data loading):

1. A: Load Command "0100 0000".
2. C: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Set BS1 to "1" and BS2 to "0". This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to "0". This selects low data byte.

Figure 119. Programming th Fuses Waveforms



mming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 239 for details on Command and Data loading):

1. A: Load Command "0010 0000".
2. C: Load Data Low Byte. Bit n = "0" programs the Lock bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

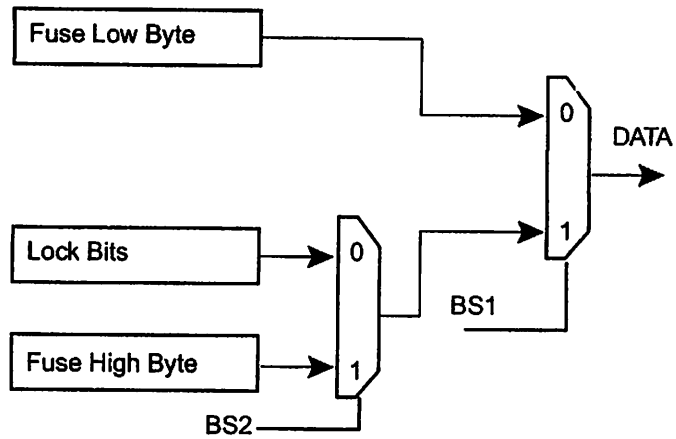
The Lock bits can only be cleared by executing Chip Erase.

g the Fuse and Lock

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 239 for details on Command loading):

1. A: Load Command "0000 0100".
2. Set $\overline{OE}$ to "0", BS2 to "0", and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
3. Set $\overline{OE}$ to "0", BS2 to "1", and BS1 to "1". The status of the Fuse High bits can now be read at DATA ("0" means programmed).
4. Set $\overline{OE}$ to "0", BS2 to "0", and BS1 to "1". The status of the Lock bits can now be read at DATA ("0" means programmed).
5. Set $\overline{OE}$ to "1".

Figure 120. Mapping Between BS1, BS2 and the Fuse- and Lock Bits During Read



j the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. A: Load Command "0000 1000".
2. B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to "0", and BS to "0". The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

j the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 239 for details on Command and Address loading):

1. A: Load Command "0000 1000".
2. B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to "0", and BS1 to "1". The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to "1".

Figure 121. Parallel Programming Timing, Including some General Timing Requirements

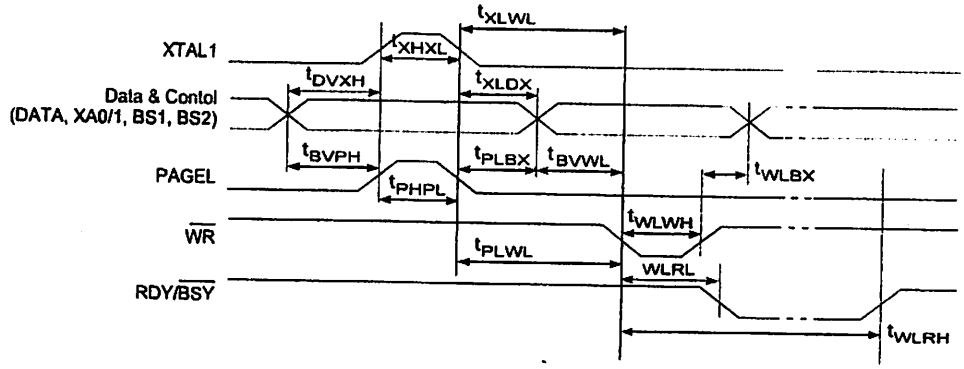
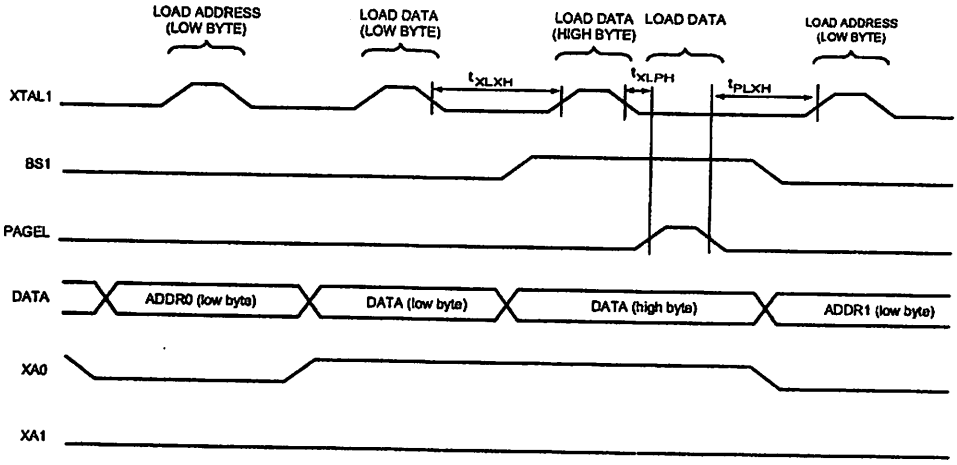
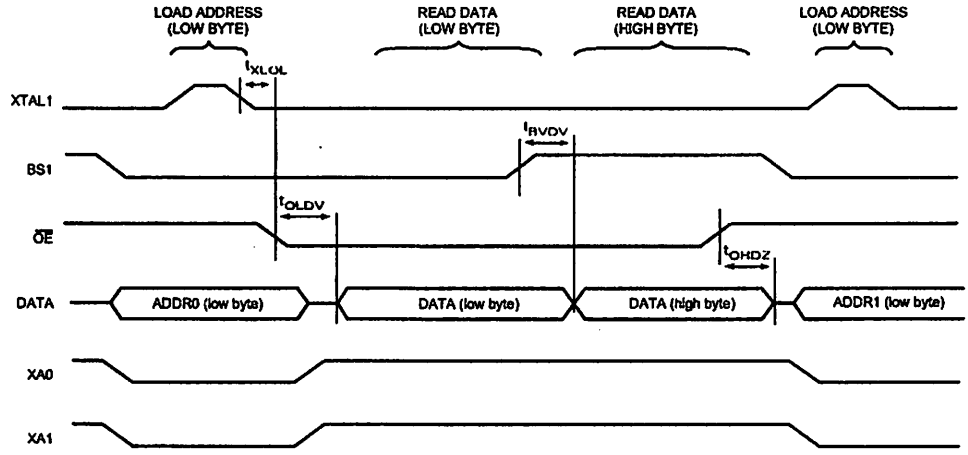


Figure 122. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 121 (i.e. t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 123. Parallel Programming Timing, Reading Sequence (within the same Page) with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 121 (i.e. t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 106. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL High	0			ns
t_{PLXH}	PAGEL low to XTAL1 High	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA Valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. t_{WLRH_CE} is valid for the Chip Erase command.

l Downloading

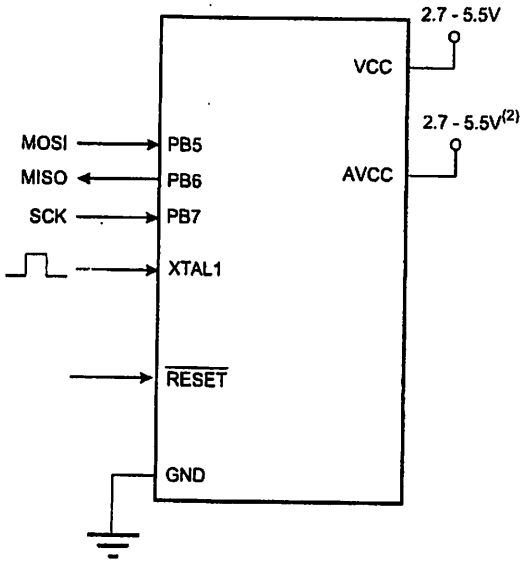
Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input), and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 107 on page 247, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Programming Pin ıg

Table 107. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial Data in
MISO	PB6	O	Serial Data out
SCK	PB7	I	Serial Clock

Figure 124. Serial Programming and Verify[1]



Notes:
1. If the device is clocked by the Internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.
2. VCC - 0.3 < AVCC < VCC + 0.3. However, AVCC should always be within 2.7 - 5.5V.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low:> 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for $f_{ck} \geq$ 12 MHz

High:> 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for $f_{ck} \geq$ 12 MHz

Programming thm

When writing serial data to the ATmega8535, data is clocked on the rising edge of SCK.

When reading data from the ATmega8535, data is clocked on the falling edge of SCK. See Figure 125 for timing details.

To program and verify the ATmega8535 in the Serial Programming mode, the following sequence is recommended (See four byte instruction formats in Table 109):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to "0". In some systems, the programmer can not guarantee that SCK is held low during Power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to "0".
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in synchronization the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The page size is found in Table 104 on page 237. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load Program Memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program Memory Page is stored by loading the Write Program Memory Page instruction with the 8 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 108.) Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 108). In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to "1".
   Turn V_{CC} power off.

lling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value 0xFF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value 0xFF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. See Table 108 for t_{WD_FLASH} value.

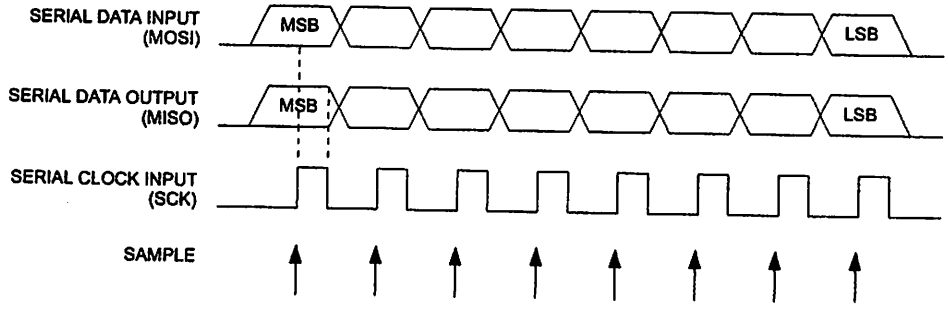
olling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value 0xFF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value 0xFF, but the user should have the following in mind: As a chip erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. This does not apply if the EEPROM is reprogrammed without chip erasing the device. In this case, data polling cannot be used for the value 0xFF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See Table 108 for t_{WD_EEPROM} value.

Table 108. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9.0 ms
t_{WD_ERASE}	9.0 ms
t_{WD_FUSE}	4.5 ms

Figure 125. Serial Programming Waveforms



09. Serial Programming Instruction Set

ess high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care

ction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
mming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
rase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Program Memory	0010 H000	0000 aaaa	bbbb bbbb	oooo oooo	Read H (high or low) data **o** from Program memory at word address **a:b**.
rogram Memory	0100 H000	0000 xxxx	xxxb bbbb	iiii iiii	Write H (high or low) data **i** to Program Memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
rogram Memory	0100 1100	0000 aaaa	bbbx xxxx	xxxx xxxx	Write Program Memory Page at address **a:b**.
EPROM Memory	1010 0000	00xx xxxa	bbbb bbbb	oooo oooo	Read data **o** from EEPROM memory at address **a:b**.
EPROM Memory	1100 0000	00xx xxxa	bbbb bbbb	iiii iiii	Write data **i** to EEPROM memory at address **a:b**.
ock Bits	0101 1000	0000 0000	xxxx xxxx	xxoo oooo	Read Lock bits. "0" = programmed, "1" = unprogrammed. See Table 96 on page 233 for details.
ock Bits	1010 1100	111x xxxx	xxxx xxxx	11ii iiii	Write Lock bits. Set bits = "0" to program Lock bits. See Table 96 on page 233 for details.
ignature Byte	0011 0000	00xx xxxx	xxxx xxbb	oooo oooo	Read Signature Byte **o** at address **b**.
use Bits	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Set bits = "0" to program, "1" to unprogram. See Table 99 on page 235 for details.
use High Bits	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Set bits = "0" to program, "1" to unprogram. See Table 98 on page 234 for details.
use Bits	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Read Fuse bits. "0" = programmed, "1" = unprogrammed. See Table 99 on page 235 for details.
use High Bits	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Read Fuse high bits. "0" = programmed, "1" = unprogrammed. See Table 98 on page 234 for details.
alibration Byte	0011 1000	00xx xxxx	0000 00bb	oooo oooo	Read Calibration Byte

al Programming aristics For characteristics of the SPI module, see "SPI Timing Characteristics" on page 256.

trical Characteristics

lute Maximum Ratings*

ting Temperature	-55°C to +125°C
ıe Temperature	-65°C to +150°C
e on any Pin except $\overline{\text{RESET}}$ spect to Ground	-1.0V to V_{CC}+0.5V
e on $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +13.0V
um Operating Voltage	6.0V
rrent per I/O Pin	40.0 mA
rrent 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.

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

ol	Parameter	Condition	Min	Typ	Max	Units
	Input Low Voltage	Except XTAL1 pin	-0.5		0.2 V_{CC}[1]	V
	Input Low Voltage	XTAL1 pin, External Clock Selected	-0.5		0.1 V_{CC}[1]	V
	Input High Voltage	Except XTAL1 and $\overline{\text{RESET}}$ pins	0.6 V_{CC}[2]		V_{CC} + 0.5	V
	Input High Voltage	XTAL1 pin, External Clock Selected	0.8 V_{CC}[2]		V_{CC} + 0.5	V
	Input High Voltage	$\overline{\text{RESET}}$ pin	0.9 V_{CC}[2]		V_{CC} + 0.4	V
	Output Low Voltage[3] (Ports A,B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
	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
	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
	Reset Pull-up Resistor		20		100	kΩ
	I/O Pin Pull-up Resistor		20		100	kΩ
	Power Supply Current	Active 4 MHz, V_{CC} = 3V (ATmega8535L)		4		mA
		Active 8 MHz, V_{CC} = 5V (ATmega8535)		14		mA
		Idle 4 MHz, V_{CC} = 3V (ATmega8535L)		2		mA
		Idle 8 MHz, V_{CC} = 5V (ATmega8535)		8		mA
	Power-down mode[5]	WDT enabled, V_{CC} = 3V		< 10		µA
		WDT disabled, V_{CC} = 3V		< 5		µA

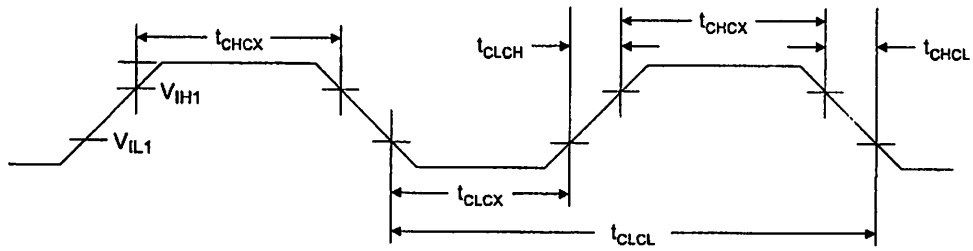
haracteristics T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

ol	Parameter	Condition	Min	Typ	Max	Units
	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			40	mV
	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA
	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

1. "Max" means the highest value where the pin is guaranteed to be read as low.
2. "Min" means the lowest value where the pin is guaranteed to be read as high.
3. Although each I/O port can sink more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP Package:
   1] The sum of all IOL, for all ports, should not exceed 200 mA.
   2] The sum of all IOL, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0 - B7,C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA.
   TQFP 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, should not exceed 100 mA.
   3] The sum of all IOL, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all IOL, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all IOL, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all IOL, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all IOL, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOL, for ports D4 - D7, should not exceed 100 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.
4. Although each I/O port can source more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP Package:
   1] The sum of all IOH, for all ports, should not exceed 200 mA.
   2] The sum of all IOH, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0 - B7,C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA.
   TQFP 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, should not exceed 100 mA.
   3] The sum of all IOH, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all IOH, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all IOH, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all IOH, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all IOH, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all IOH, for ports D4 - D7, should not exceed 100 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.
5. Minimum V_{CC} for Power-down is 2.5V.

nal Clock Drive forms

Figure 126. External Clock Drive Waveforms



nal Clock Drive

Table 110. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.5V to 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	8	0	16	MHz
t_{CLCL}	Clock Period	125		62.5		ns
t_{CHCX}	High Time	50		25		ns
t_{CLCX}	Low Time	50		25		ns
t_{CLCH}	Rise Time		1.6		0.5	µs
t_{CHCL}	Fall Time		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2	%

Table 111. External RC Oscillator, Typical Frequencies[1]

R [kΩ]	C [pF]	f
100	70	TBD
31.5	20	TBD
6.5	20	TBD

Note: 1. R should be in the range 3 kΩ - 100 kΩ, and C should be at least 20 pF. The C values given in the table includes pin capacitance. This will vary with package type.

/ire Serial Interface Characteristics

12 describes the requirements for devices connected to the Two-wire Serial Bus. The ATmega8535 Two-wire Serial e meets or exceeds these requirements under the noted conditions.

symbols refer to Figure 127.

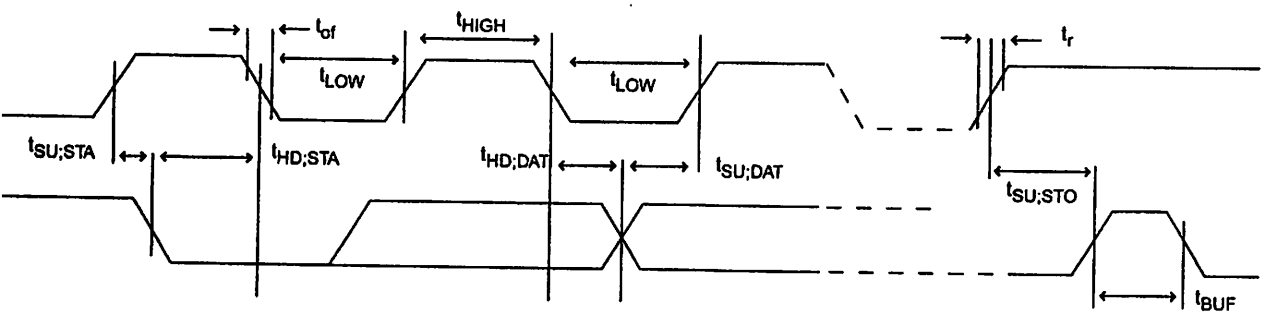
12. Two-wire Serial Bus Requirements

ol	Parameter	Condition	Min	Max	Units
	Input Low Voltage		-0.5	$0.3\ V_{CC}$	V
	Input High Voltage		$0.7\ V_{CC}$	$V_{CC} + 0.5$	V
	Hysteresis of Schmitt Trigger Inputs		$0.05\ V_{CC}$(2)	–	V
	Output Low Voltage	3 mA sink current	0	0.4	V
	Rise Time for both SDA and SCL		$20 + 0.1C_b$(3)(2)	300	ns
	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF(3)	$20 + 0.1C_b$(3)(2)	250	ns
	Spikes Suppressed by Input Filter		0	50(2)	ns
	Input Current each I/O Pin	$0.1V_{CC} < V_I < 0.9V_{CC}$	-10	10	µA
	Capacitance for each I/O Pin		–	10	pF
	SCL Clock Frequency	f_{CK}(4) > max(16f_{SCL}, 250kHz)(5)	0	400	kHz
	Value of Pull-up resistor	$f_{SCL} \le 100$ kHz	$\frac{V_{CC} - 0.4V}{3mA}$	$\frac{1000ns}{C_b}$	Ω
		$f_{SCL} > 100$ kHz	$\frac{V_{CC} - 0.4V}{3mA}$	$\frac{300ns}{C_b}$	Ω
	Hold Time (Repeated) START Condition	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
	Low Period of the SCL Clock	$f_{SCL} \le 100$ kHz(6)	4.7	–	µs
		$f_{SCL} > 100$ kHz(7)	1.3	–	µs
	High Period of the SCL clock	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
	Set-up Time for a Repeated START Condition	$f_{SCL} \le 100$ kHz	4.7	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
	Data hoLd Time	$f_{SCL} \le 100$ kHz	0	3.45	µs
		$f_{SCL} > 100$ kHz	0	0.9	µs
	Data Setup Time	$f_{SCL} \le 100$ kHz	250	–	ns
		$f_{SCL} > 100$ kHz	100	–	ns
	Setup Time for STOP Condition	$f_{SCL} \le 100$ kHz	4.0	–	µs
		$f_{SCL} > 100$ kHz	0.6	–	µs
	Bus Free Time between a STOP and START Condition	$f_{SCL} \le 100$ kHz	4.7	–	µs
		$f_{SCL} > 100$ kHz	1.3	–	µs

. In ATmega8535, this parameter is characterized and not 100% tested.
. Required only for $f_{SCL} > 100$ kHz.
. C_b = capacitance of one bus line in pF.
. f_{CK} = CPU clock frequency.

5. This requirement applies to all ATmega8535 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the ATmega8535 Two-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus f_{CK} must be greater than 6 MHz for the low time requirement to be strictly met at f_{SCL} = 100 kHz.
7. The actual low period generated by the ATmega8535 Two-wire Serial Interface is ($1/f_{SCL} - 2/f_{CK}$), thus the low time requirement will not be strictly met for f_{SCL} > 308 kHz when f_{CK} = 8 MHz. Still, ATmega8535 devices connected to the bus may communicate at full speed (400 kHz) with other ATmega8535 devices, as well as any other device with a proper t_{LOW} acceptance margin.

127. Two-wire Serial Bus Timing



See Figure 128 and Figure 129 for details.

Table 113. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 59		ns
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		$5 \cdot t_{SCK}$		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	$4 \cdot t_{ck}$			
11	SCK high/low	Slave	$2 \cdot t_{ck}$			
12	Rise/Fall time	Slave		TBD		
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	SS low to SCK	Slave	$2 \cdot t_{ck}$			

Figure 128. SPI Interface Timing Requirements (Master Mode)

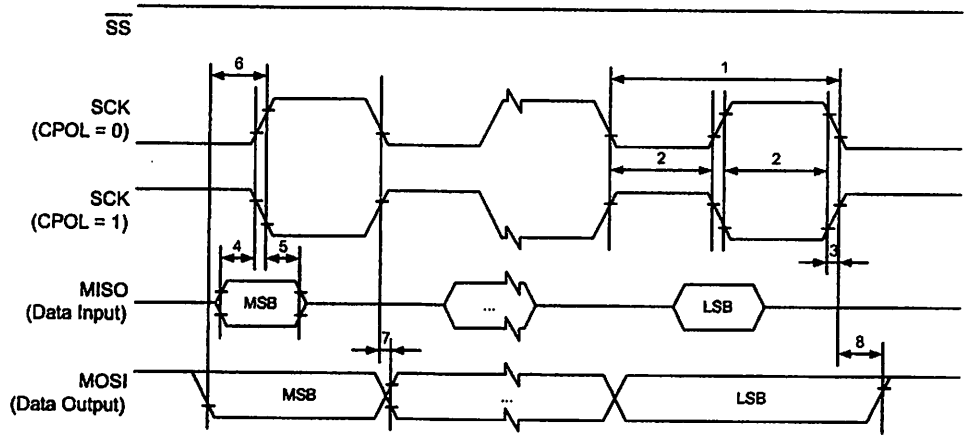
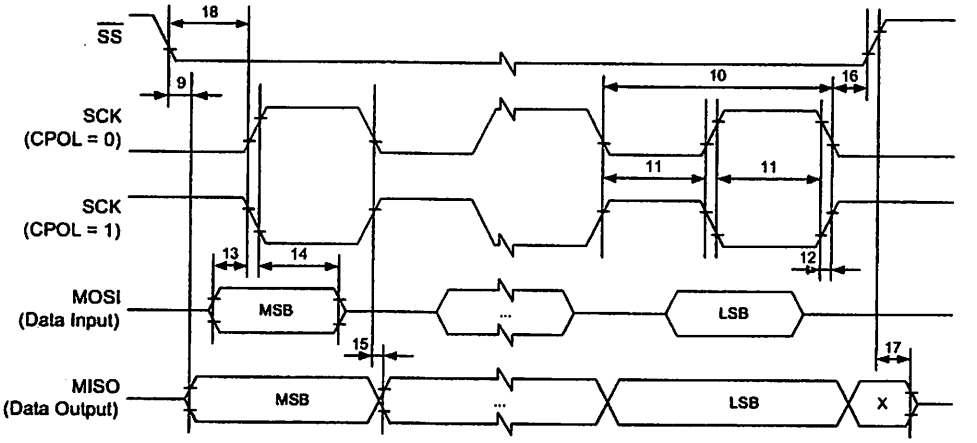


Figure 129. SPI Interface Timing Requirements (Slave Mode)



Characteristics – Preliminary Data

14. ADC Characteristics

ol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution	Single Ended Conversion		10		Bits
		Differential Conversion Gain = 1x or 20x		8		Bits
		Differential Conversion Gain = 200x		7		Bits
	Absolute Accuracy	Single Ended Conversion V_{REF} = 4V ADC clock = 200 kHz		1	TBD	LSB
	Integral Non-linearity	V_{REF} = 4V		0.5		LSB
	Differential Non-linearity	V_{REF} = 4V		0.5		LSB
	Zero Error (Offset)	V_{REF} = 4V		1		LSB
	Conversion Time	Free Running Conversion	65		260	µs
	Clock Frequency		50		200	kHz
	Analog Supply Voltage		V_{CC} - 0.3(2)		V_{CC} + 0.3(3)	V
	Reference Voltage	Single Ended Conversion	2.0		AV_{CC}	V
		Differential Conversion	2.0		$AV_{CC-0.2}$	V
	Input Voltage	Single ended channels	GND		V_{REF}	
		Differential channels	TBD		TBD	
	Input Bandwidth	Single ended channels		TBD		kHz
		Differential channels		4		kHz
	Internal Voltage Reference		TBD	2.56	TBD	V
	Reference Input Resistance		TBD	TBD	TBD	kΩ
	Analog Input Resistance			TBD		MΩ

1. Values are guidelines only. Actual values are TBD.
2. Minimum for AV_{CC} is 2.7V.
3. Maximum for AV_{CC} is 5.5V.

ega8535 Typical acteristics – ninary Data

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: Operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as C_L*V_{CC}*f where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

ster Summary

ss	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
5F)	SREG	I	T	H	S	V	N	Z	C	8
5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	10
5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
5C)	OCR0	Timer/Counter0 Output Compare Register								82
5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	47, 66
5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	67
59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	82, 112, 130
58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	83, 113, 131
57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	224
56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	177
55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	30, 65
54)	MCUCSR	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF	38, 66
53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	80
52)	TCNT0	Timer/Counter0 (8 Bits)								82
51)	OSCCAL	Oscillator Calibration Register								28
50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	56,85,132,199,219
4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	107
4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	110
4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								111
4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								111
4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								111
4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								111
49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								111
48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								111
47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								111
46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								111
45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	125
44)	TCNT2	Timer/Counter2 (8 Bits)								127
43)	OCR2	Timer/Counter2 Output Compare Register								128
42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	128
41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	40
40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				165
40)[1]	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	163
3F)	EEARH	–	–	–	–	–	–	–	EEAR8	17
3E)	EEARL	EEPROM Address Register Low Byte								17
3D)	EEDR	EEPROM Data Register								17
3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	17
3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	63
3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	63
39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	63
38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	63
37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	63
36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	64
35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	64
34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	64
33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	64
32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	64
31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	64
30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	64
2F)	SPDR	SPI Data Register								139
2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	139
2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	137
2C)	UDR	USART I/O Data Register								160
2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	161
2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	162
29)	UBRRL	USART Baud Rate Register Low Byte								165
28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	199
27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	215
26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	217
25)	ADCH	ADC Data Register High Byte								218
24)	ADCL	ADC Data Register Low Byte								218
23)	TWDR	Two-wire Serial Interface Data Register								179
22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	180
21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	179

ster Summary (Continued)

ess	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
x20)	TWBR	Two-wire Serial Interface Bit Rate Register								177

1. Refer to the USART description for details on how to access UBRRH and UCSRC.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

uction Set Summary

nics	Operands	Description	Operation	Flags	#Clocks
TIC AND LOGIC INSTRUCTIONS					
	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
	Rd	One's Complement	Rd ← 0xFF − Rd	Z,C,N,V	1
	Rd	Two's Complement	Rd ← 0x00 − Rd	Z,C,N,V,H	1
	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
	Rd	Decrement	Rd ← Rd − 1	Z,N,V	1
	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
	Rd	Set Register	Rd ← 0xFF	None	1
	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
NSTRUCTIONS					
	k	Relative Jump	PC ← PC + k + 1	None	2
		Indirect Jump to (Z)	PC ← Z	None	2
	k	Direct Jump	PC ← k	None	3
	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
		Indirect Call to (Z)	PC ← Z	None	3
	k	Direct Subroutine Call	PC ← k	None	4
		Subroutine Return	PC ← STACK	None	4
		Interrupt Return	PC ← STACK	I	4
	Rd,Rr	Compare, Skip if Equal	If (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
	Rd,Rr	Compare	Rd − Rr	Z, N,V,C,H	1
	Rd,Rr	Compare with Carry	Rd − Rr − C	Z, N,V,C,H	1
	Rd,K	Compare Register with Immediate	Rd − K	Z, N,V,C,H	1
	Rr, b	Skip if Bit in Register Cleared	If (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
	Rr, b	Skip if Bit in Register is Set	If (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
	P, b	Skip if Bit in I/O Register Cleared	If (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
	P, b	Skip if Bit in I/O Register is Set	If (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
	s, k	Branch if Status Flag Set	If (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
	s, k	Branch if Status Flag Cleared	If (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
	k	Branch if Equal	If (Z = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Not Equal	If (Z = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Carry Set	If (C = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Carry Cleared	If (C = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Same or Higher	If (C = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Lower	If (C = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Minus	If (N = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Plus	If (N = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Greater or Equal, Signed	If (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Less Than Zero, Signed	If (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Half Carry Flag Set	If (H = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Half Carry Flag Cleared	If (H = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if T Flag Set	If (T = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if T Flag Cleared	If (T = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Overflow Flag is Set	If (V = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Overflow Flag is Cleared	If (V = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Interrupt Enabled	If (I = 1) then PC ← PC + k + 1	None	1 / 2

nics	Operands	Description	Operation	Flags	#Clocks
	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
ANSFER INSTRUCTIONS					
	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
	Rd, K	Load Immediate	Rd ← K	None	1
	Rd, X	Load Indirect	Rd ← (X)	None	2
	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
	Rd, Y	Load Indirect	Rd ← (Y)	None	2
	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
	Rd, Z	Load Indirect	Rd ← (Z)	None	2
	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
	X, Rr	Store Indirect	(X) ← Rr	None	2
	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
	Y, Rr	Store Indirect	(Y) ← Rr	None	2
	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
	Z, Rr	Store Indirect	(Z) ← Rr	None	2
	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
		Load Program Memory	R0 ← (Z)	None	3
	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
		Store Program Memory	(Z) ← R1:R0	None	-
	Rd, P	In Port	Rd ← P	None	1
	P, Rr	Out Port	P ← Rr	None	1
	Rr	Push Register on Stack	STACK ← Rr	None	2
	Rd	Pop Register from Stack	Rd ← STACK	None	2
IT-TEST INSTRUCTIONS					
	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
	s	Flag Set	SREG(s) ← 1	SREG(s)	1
	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
		Set Carry	C ← 1	C	1
		Clear Carry	C ← 0	C	1
		Set Negative Flag	N ← 1	N	1
		Clear Negative Flag	N ← 0	N	1
		Set Zero Flag	Z ← 1	Z	1
		Clear Zero Flag	Z ← 0	Z	1
		Global Interrupt Enable	I ← 1	I	1
		Global Interrupt Disable	I ← 0	I	1
		Set Signed Test Flag	S ← 1	S	1
		Clear Signed Test Flag	S ← 0	S	1
		Set Twos Complement Overflow.	V ← 1	V	1
		Clear Twos Complement Overflow	V ← 0	V	1
		Set T in SREG	T ← 1	T	1
		Clear T in SREG	T ← 0	T	1
		Set Half Carry Flag in SREG	H ← 1	H	1
		Clear Half Carry Flag in SREG	H ← 0	H	1

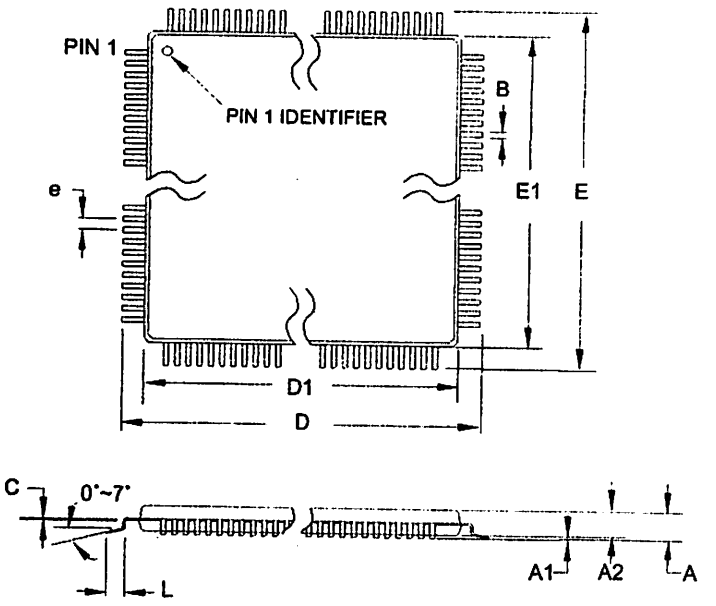
ıics	Operands	Description	Operation	Flags	#Clocks
TROL INSTRUCTIONS					
		No Operation		None	1
		Sleep	(see specific descr. for Sleep function)	None	1
		Watchdog Reset	(see specific descr. for WDR/Timer)	None	1
		Break	For On-chip Debug Only	None	N/A

ring Information[1]

eed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega8535L-8AC ATmega8535L-8PC ATmega8535L-8JC ATmega8535L-8MC	44A 40P6 44J 44M1	Commercial (0°C to 70°C)
		ATmega8535L-8AI ATmega8535L-8PI ATmega8535L-8JI ATmega8535L-8MI	44A 40P6 44J 44M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega8535-16AC ATmega8535-16PC ATmega8535-16JC ATmega8535-16MC	44A 40P6 44J 44M1	Commercial (0°C to 70°C)
		ATmega8535-16AI ATmega8535-16PI ATmega8535-16JI ATmega8535-16MI	44A 40P6 44J 44M1	Industrial (-40°C to 85°C)

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

	Package Type
	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
	44-lead, Plastic J-ledded Chip Carrier (PLCC)
	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Micro Lead Frame Package (MLF)



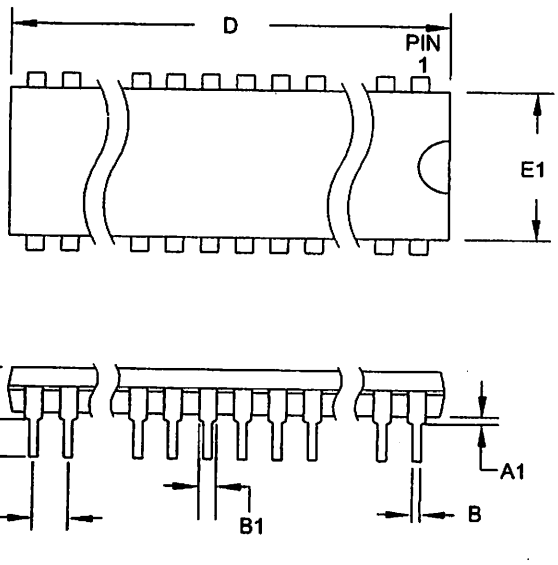
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

otes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B



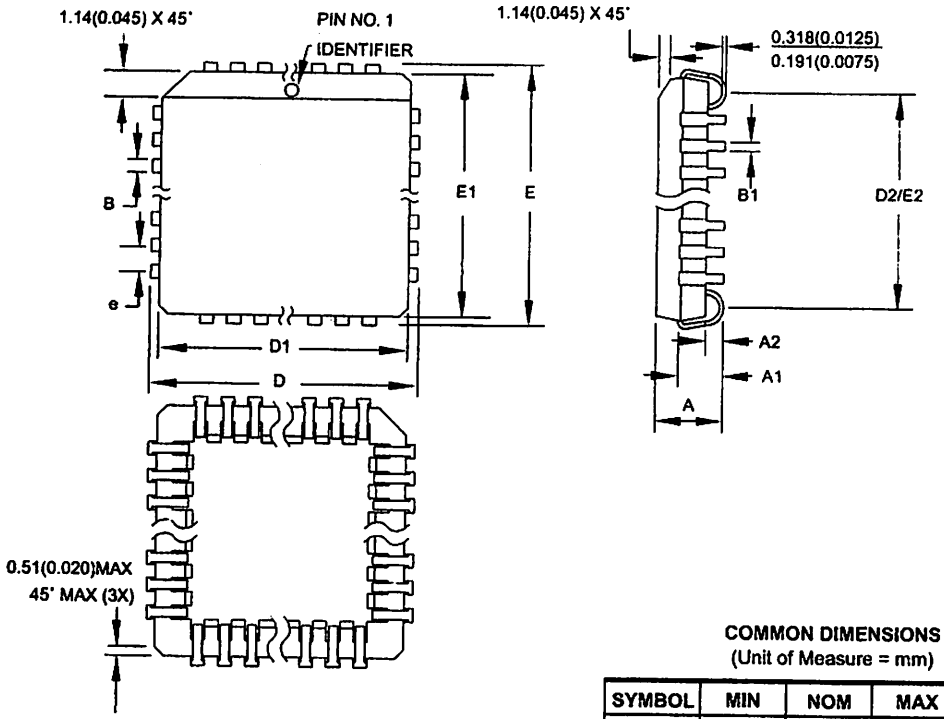
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B



COMMON DIMENSIONS
(Unit of Measure = mm)

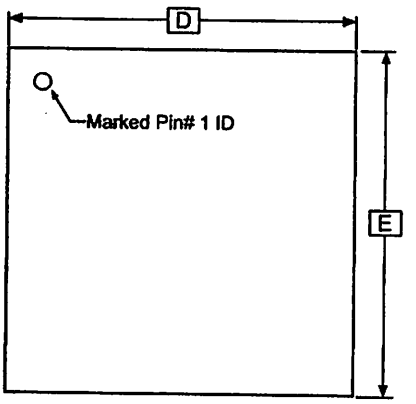
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

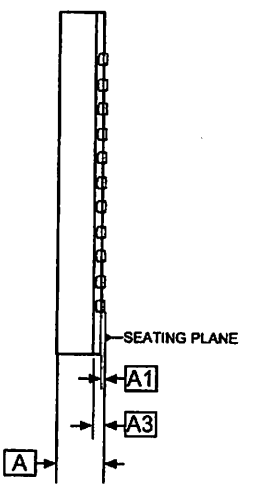
10/04/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B

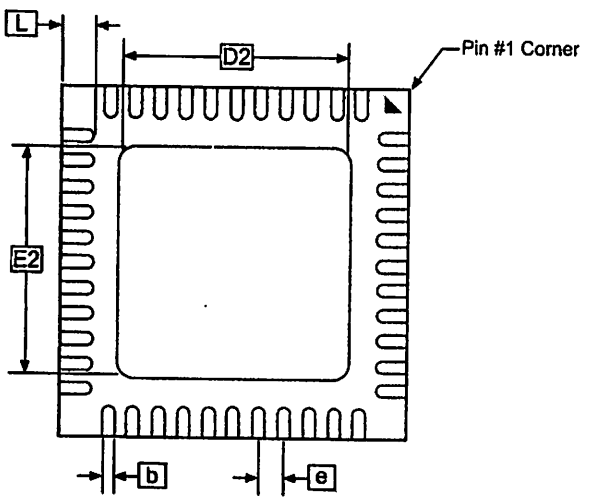
A



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.25 REF		
b	0.18	0.23	0.30	
D		7.00 BSC		
D2	5.00	5.20	5.40	
E		7.00 BSC		
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.35	0.55	0.75	

Note: JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-1.

01/15/03

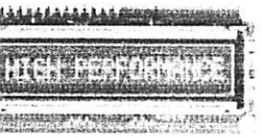
	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44M1**, 44-pad, 7 x 7 x 1.0 mm Body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)	44M1	C

Dot Matrix Liquid Crystal Display Modules

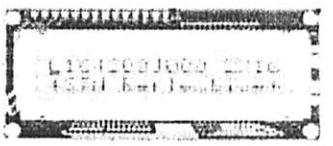
HARACTER TYPE

• FEATURES :

- Slim, light weight and low power consumption
- High contrast and wide viewing angle
- Built-in controller for easy interfacing
- LCD modules with built-in EL or LED backlight



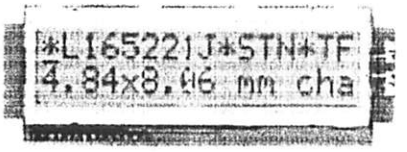
M1641



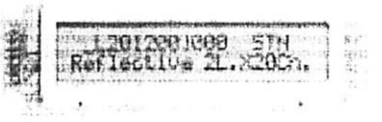
L1642

L1614

M1632



L1652



L2012

• SPECIFICATIONS :

: Standard products : Products of optional specification

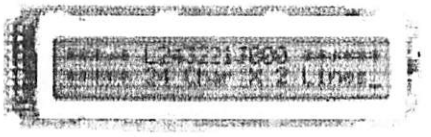
ter Format (character x line)		16 x 1	16 x 2	16 x 2	16 x 2	16 x 4	20 x 2
		M1641	M1632	L1642	L1652	L1614	L2012
ve		M16410AS	M16320AS	L164200J000S	L165200J200S	L161400J000S	L201200J000S
light		M16419DWS	M16329DWS	L164221J000S	L165221J200S	L161421J000S	L201221J000S
cklight		M16417DYS	M16327DYS	L1642B1J000S	L1652B1J200S	L1614B1J000S	L2012B1J000S
ve (wide temp)		M16410CS	M16320CS	L164200L000S	L165200L200S	L161400L000S	L201200L000S
cklight (wide temp)		M16417JYS	M16327JYS	L1642B1L000S	L1652B1L200S	L1614B1L000S	L2012B1L000S
er font		5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
	Reflective	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
	EL backlight	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
) mm	LED backlight	80,0 x 36,0 x 15,8	80,0 x 30,0 x 15,8	80,0 x 36,0 x 15,8	122,0 x 44,0 x 15,8	87,0 x 60,0 x 15,8	116,0 x 37,0 x 15,8
area (HxV) mm		64,5 x 13,8	62,0 x 16,0	64,5 x 13,8	99,0 x 24,0	61,8 x 25,2	83,0 x 18,6
er size (HxV) mm *1		3,07 x 5,73	2,78 x 4,27	2,95 x 3,80	4,84 x 8,06	2,95 x 4,15	3,20 x 4,85
(HxV) mm		0,55 x 0,75	0,50 x 0,55	0,50 x 0,55	0,92 x 1,10	0,55 x 0,55	0,60 x 0,65
upply voltage (VDD-VSS) V		+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
consumption	IDD	1,5	2,0	1,6	2,0	2,7	2,0
)	ILC *4	0,2	0,2	0,3	0,4	1,1	0,4
method (duty)		1/16	1/16	1/16	1/16	1/16	1/16
LSI		KS0066 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent
g temperature (°C)	normal temp.	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50
	wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
temperature (°C)	normal temp.	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
	wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
Reflective		25	25	25	50	50	40
EL backlight		30	30	30	55	55	45
LED backlight		35	40	35	65	65	60
Model		5S	5S	5S	5C	5A	5A
Power supply (V)		+ 5.0	+ 5,0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
current consumption (mA) *3		10	10	10	35	45	45
Forward current consumption (mA)		100	112	100	240	200	154
Forward input voltage (V.typ.)		+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

ding cursor
external temperature compensation
ding EL backlight
l on normal temperature range

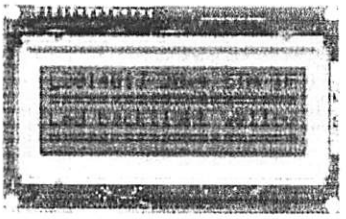
H : Horizontal V : Vertical T : Thickness (max)

policy is one of continues improvements we reserve the right to change the specifications for the products in the catalogue without notice.

L2022

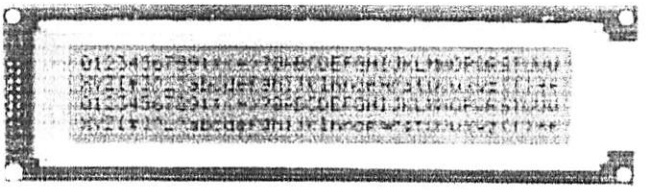


L2432



L2014

L4042



M4024

• SPECIFICATIONS :

[ ] : Standard products [ ] : Products of optional specification

Character Format (character x line)			20 x 2	20 x 4	24 x 2	40 x 2	40 x 4
Model			L2022	L2014	L2432	L4042	M4024
Reflective			-	L201400J000S	L243200J000S	L404200J000S	M40240AS
EL backlight			-	L201421J000S	L243221J000S	L404221J000S	M40249DWS
LED backlight			-	L2014B1J000S	L2432B1J000S	L4042B1J000S	M40247DYS
Reflective (wide temp)			L202200P000S	L201400L000S	L243200L000S	L404200L000S	M40240CS
LED backlight (wide temp)			L2022B1P000S	L2014B1L000S	L2432B1L000S	L4042B1L000S	M40247JYS
Character font			5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
Module size (HxVxT) mm	Reflective		180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
	EL backlight		180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
	LED backlight		180,0 x 40,0 x 14,8	98,0 x 60,0 x 15,8	118,0 x 36,0 x 15,8	182,0 x 33,5 x 16,3	190,0 x 54,0 x 16,3
Viewing area (HxV) mm			149,0 x 23,0	76,0 x 25,2	94,5 x 17,8	154,4 x 15,8	147,0 x 29,5
Character size (HxV) mm *1			6,00 x 9,66	2,95 x 4,15	3,20 x 4,85	3,20 x 4,85	2,78 x 4,27
Dot size (HxV) mm			1,12 x 1,12	0,55 x 0,55	0,60 x 0,65	0,60 x 0,65	0,50 x 0,55
Power supply voltage (VDD-VSS) V			+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
Current consumption (mA,typ)		IDD	4,2	2,9	2,5	3,0	8,0
		ILC *4	2,6	1,2	0,5	1,0	3,0
Driving method (duty)			1/16	1/16	1/16	1/16	1/16
Built-in LSI			KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent
Operating temperature (°C)		normal temp.	-	0 to + 50	0 to + 50	0 to + 50	0 to + 50
		wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
Storage temperature (°C)		normal temp.	-	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
		wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
Weight (g, typ.)	Reflective		80	55	40	70	90
	EL backlight		-	60	45	75	105
	LED backlight		110	70	60	95	140
Inverters for EL	Model		-	5A	5A	5C	5D
	Power supply (V)		+ 5.0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
	current consumption (mA) *3		-	45	45	25	80
LED backlight	Forward current consumption (mA)		320	240	150	260	480
	Forward input voltage (V,typ.)		+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

*1 : Excluding cursor H : Horizontal V : Vertical T : Thickness (max)

*2 : With external temperature compensation

*3 : Including EL backlight

*4 : Based on normal temperature range

)ot Matrix Liquid Crystal Display Modules

RAPHIC TYPE

• FEATURES :

- •Wide viewing angle and high contrast
- •Full dot configuration fits any application
- •Slim, light weight and low power consumption
- •Available in STN and FSTN

• SPECIFICATIONS :

at (HxV,dot)			97 x 32	128 x 32	128 x 64	128 x 64
			Y97031	G1213	G1216	G1226
e	Reflective	built-in RAM	-	-	-	-
ode)	Reflective wide temp.	built-in RAM	-	G121300N000S	G121600N000S	-
	LED backlight	built-in RAM	-	-	-	G1226B1J000S
	LED backlight wide temp	built-in RAM	-	G1213B1N000S	G1216B1N000S	-
pe	Transmissive	-	-	-	-	-
ode)	with CFL backlight	built-in controller	-	-	-	-
	Transflective	built-in RAM	Y97031LF60W	-	-	-
ize	Reflective (no backlight)		47,5 x 65,4 x 2,1	75,0 x 41,5 x 6,8	75,0 x 52,7 x 6,8	-
T)	LED backlight		-	75,0 x 41,5 x 8,9	75,0 x 52,7 x 8,9	93,0 x 70,0 x 11,4
	CFL backlight		-	-	-	-
area (HxV) mm			43,5 x 23,9	60,0 x 21,3	60,0 x 32,5	70,7 x 38,8
(H x V) mm			0,35 x 0,48	0,40 x 0,48	0,40 x 0,40	0,44 x 0,44
(H x V) mm			0,39 x 0,52	0,43 x 0,51	0,43 x 0,43	0,48 x 0,48
upply voltage (V)		(VDD - VSS)	+ 5,0	+ 5,0	+ 5,0	+ 5,0
		(VLC - VSS)	-	- 8,0	- 8,1	-8,2
onsumption		IDD	0,10	2,0	2,0	3,0
		IDD (built-in controller)	-	-	-	-
)		ILC	-	1,8	1,8	2,0
Driving method (duty)			1/33	1/64	1/64	1/64
SI		Driver	SED1530 or equivalent	HD61202 HD61203 or equivalent	HD61202 HD61203 or equivalent	KS0107 KS0108 or equivalent
		Controller	-	-	-	-
g temperature range (°C)			- 20 to + 70	- 20 to + 70	- 20 to + 70	0 to + 50
emperature range (°C)			- 30 to + 80	- 30 to + 80	- 30 to + 80	- 20 to + 60
	Reflective (Transflective no backlight)		10	23	35	-
	LED backlight		-	35	45	72
	CFL backlight		-	-	-	-
light	Forward current consumption (mA)		-	40	90	125
	Forward input voltage (V, typ.)		-	3,8	4,1	4,1
	Mode		-	-	-	-
or CFL	Power supply voltage (V)		-	-	-	-
	Current consumption (mA, typ.)		-	-	-	-

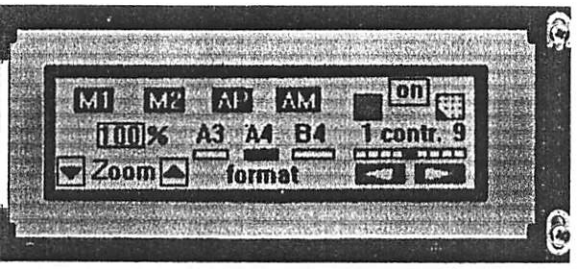
n DC/DC converter (single power source)

vith external temperature compensation circuit

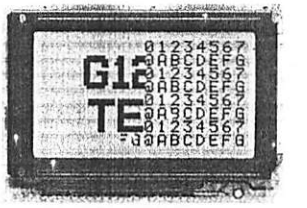
policy is one of continues improvements we reserve the right to change the specifications of the products in the catalogue without notice.

ormat (HxV,dot)			240 x 64	240 x 128	320 x 200	320 x 240	640 x 200
el			G2446	G242C	G321D	G324E	G649D
type	Reflective	built-in RAM	-	-	-	-	-
y mode)	Reflective wide temp.	built-in RAM	-	-	-	-	-
	LED backlight	built-in RAM	-	-	-	-	-
	LED backlight wide temp	built-in RAM	-	-	-	-	-
N type	Transmissive	-	G2446X5R1A0S	G242CX5R1ACS	G321DX5R1A0S	G324EX5R1A0S	G649DX5R010S
f mode)	with CFL backlight	built-in controller	G2446X5R1ACS	G242CX5R1A0S	G321DX5R1ACS	G324EX5R1ACS	-
	Transflective	built-in RAM	-	-	-	-	-
ule size	Reflective (no backlight)			-	-	-	-
V x T)	LED backlight			-	-	-	-
	CFL backlight		191,0 x 79,0 x 15,1	180,0 x 110,0 x 15,1	166,0 x 134,0 x 15,1	166,0 x 134,0 x 15,1	260,0 x 122,0 x 15,7
ing area (HxV) mm			134,0 x 41,0	134,0 x 76,0	128,0 x 110,0	128,0 x 110,0	216,0 x 83,0
ize (H x V) mm			0,49 x 0,49	0,47 x 0,47	0,34 x 0,48	0,32 x 0,39	0,30 x 0,36
itch (H x V) mm			0,53 x 0,53	0,51 x 0,51	0,38 x 0,52	0,36 x 0,43	0,33 x 0,39
r supply voltage (V)		(VDD - VSS)	+5,0	+5,0	+5,0	+5,0	+5,0
		(VLC - VSS)	*1	*1	-24,0	-24,0	-24,0
nt consumption		IDD	12	30	8	7,5	11
		IDD (built-in controller)	15	40	23	23	-
typ.)		ILC	-	-	6	6,5	9
	Driving method (duty)		1/64	1/128	1/200	1/240	1/200
in LSI		Driver	MSM5298 MSM5299 or equivalent	KS0103 KS0104 or equivalent	MSM5298 MSM5299 or equivalent	HD66204 HD66205 or equivalent	MSM5298 MSM5299 or equivalent
		Controller	SED1330FB	SED1330FB	SED1330FB	SED1330FB	-
ating temperature range (°C)			0 to +50	0 to +50	0 to +50	0 to +50	0 to +50
ge temperature range (°C)			- 20 to +60	- 20 to +60	- 20 to +60	- 20 to +60	- 20 to +60
t	Reflective (Transflective no backlight)		-	-	-	-	-
p.)	LED backlight		-	-	-	-	-
	CFL backlight		200	280	350	350	420
acklight	Forward current consumption (mA)		-	-	-	-	-
	Forward input voltage (V, typ.)		-	-	-	-	-
	Mode		4800210	4800210	4800210	4800210	4800120
ter for CFL	Power supply voltage (V)		+5,0	+5,0	+5,0	+5,0	+12,0
	Current consumption (mA, typ.)		250	350	365	365	390

uilt-in DC/DC converter (single power source)
se with external temperature compensation
our policy is one of continous improvemets, we reserve the right to change the specifications of the products in the catalogue without notice.

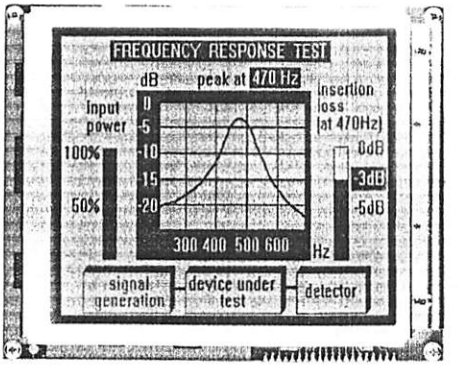


G2446



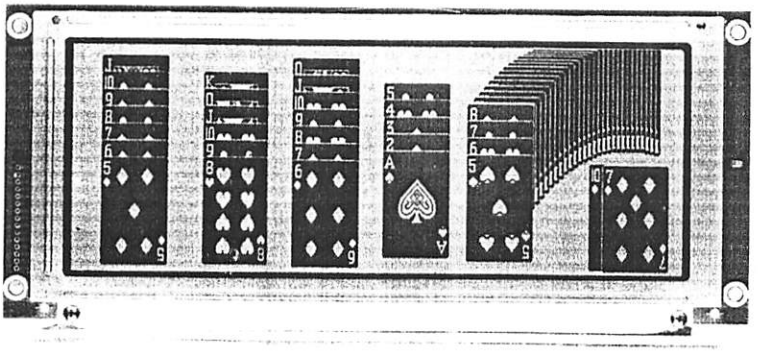
G1226

G1216

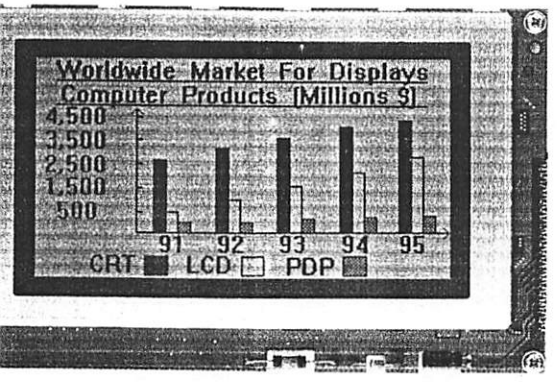


G321D

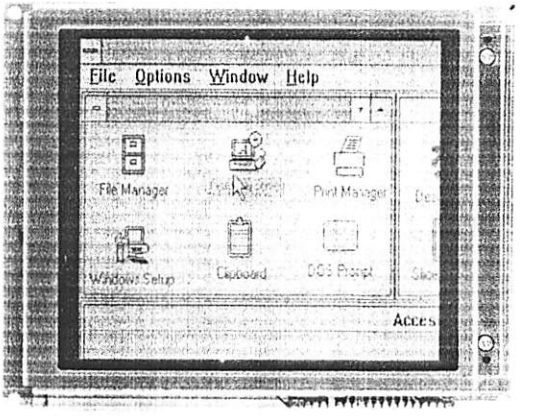
G1213



G649D



G242C



G324E

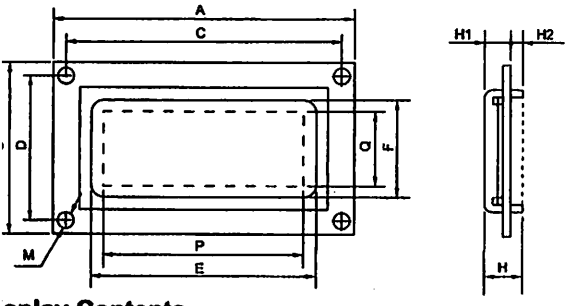
ECK LIST FOR CUSTOM DESIGNED LCD MODULE

ompany_____ **2. Application**_____ **3. Customer Specified Part No.**_____

esign

New ☐Modified : Manufacturer_____ , Part No._____ , Remarks _____
Equivalent: Manufacturer_____ , Part No._____ , Remarks _____

CM Dimensions



A x B : Module size _____ x _____ mm
E x F : Viewing area _____ x _____ mm
P x Q : Active display area _____ x _____ mm
C : Length between mounting holes _____ mm
D : Length between mounting holes _____ mm
M : Diameter of mounting hole _____ mm
H : Total thickness _____ mm
H1 : Upper thickness _____ mm
H2 : Lower thickness _____ mm

splay Contents

Character type:_____characters_____lines
Character font_____x_____dots + cursor
Character pitch_____x_____mm
Dot pitch_____x_____mm
Dot size_____x_____mm
Graphics (Full dot) type:_____x_____dots
Dot pitch_____x_____mm
Dot size_____x_____mm
Segment type:_____digits_____lines
Others_____

CD Panel

wing angle: ☐6 o'clock ☐12 o'clock☐____o'clock
pe: ☐TN ☐FSTN (Black and white)
☐STN (☐Yellow green ☐Gray ☐Blue)
Chromaticity coordinates
(_____ ≤ x ≤ _____ , _____ ≤ y ≤ _____)
☐Positive type ☐Negative type
☐Reflective ☐Transflective ☐Transmissive
☐Others_____
ıy scale: ☐Yes_____ gray scale ☐No
ferential specifications:
☐Response time t_{on} ___ ms (__ °C) t_{off} ___ ms (__ °C)
☐Viewing angle ___ deg. (__ °C) ☐Contrast ___ (__ °C)
☐Others_____
D surface finishing:
☐Normal ☐Anti-glare☐_____
arizer color: ☐Normal (neutral gray) ☐Red
☐Green ☐Blue☐_____

iving Method

iplexing:1/_____ duty, 1/_____ bias
ne frequency: _____ Hz

D driver: ☐Specified ☐Unspecified
Segment driver _____(Manufacturer _____)
Common driver _____(Manufacturer _____)
ıtroller: ☐Internal ☐External
ype No._____(Manufacturer _____)
U: ☐Internal ☐External
ype No._____(Manufacturer _____)
M: ☐Internal ☐External
/pe No. /Memory size____(Kbit) (Manufacturer _____)

ower Supply

ingle power supply: ☐5V ☐_____V
power supplies
For logic: (V_{DD}-V_{SS}) : ☐5V ☐_____V
For LC drive: (V_{LC}-V_{SS}) : ☐_____V

11. Temperature Compensation Circuit

☐Internal ☐External ☐Unnecessary
Compensation range: ☐ 0°C to 50°C ☐ __°C to __°C

12. Current Consumption

For logic: typ._____mA, max._____mA
For LC drive: typ._____mA, max._____mA
Others ( ) : typ._____mA, max._____mA

13. Contrast Adjustment

☐Internal ☐External ☐Unnecessary
Method: ☐Temp. compensation circuit ☐Volume ☐____

14. Temperature Range

Operating temperature range: ☐ 0°C to 50°C ☐____°C to____°C
Storage temperature range: ☐- 20°C to 60°C ☐____°C to____°C

15. Input/Output Terminals

Specifying allocation: ☐Yes ☐No
Specifying position: ☐Yes ☐No

16. Weight

typ.____ g, max.____ g

17. Connector

☐ Internal ☐ External ☐ Unnecessary
Type No._____(Manufacturer _____)

18. Backlight

☐Internal ☐External ☐Unnecessary
☐EL: ☐Green ☐White ☐_____
☐LED: ☐Yellow green ☐Amber ☐_____
☐CFL: ☐White ☐_____
☐Incandescent lamp ☐Others_____
☐Backlight type ☐Edge backlight type
Brightness:_____cd/m²
Inverter: ☐Internal ☐External ☐Unnecessary
Power supply voltage_____V
Current consumption (backlight included)_____ mA
Brightness control: ☐Yes ☐No

19. Others

_____
_____
_____

20. Schedule

Estimate: _____
Sample: Delivery_____, Quantity:_____pcs
Mass production: Target price: _____
Delivery _____, Total quantity:_____pcs
Quantity per month_____pcs

uid Crystal Displays

:CK LIST FOR CUSTOM DESIGNED LCD

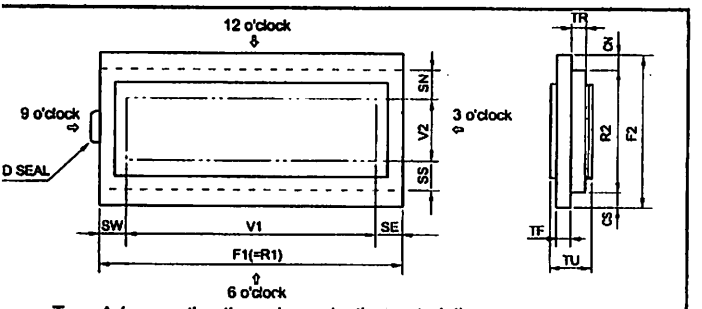
npany_____ **2. Application**_____ **3. Customer Specified Part No.**_____

sign

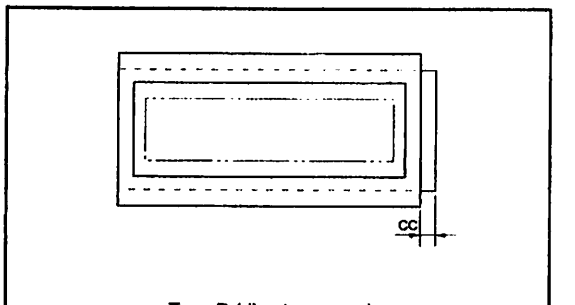
ew ☐Modified: Manufacturer _____ , Part No. _____ , Remarks _____

quivalent: Manufacturer_____ , Part No. _____ , Remarks_____

iel Dimensions



- Type A (connection through conductive material) -



- Type B (direct common) -

rizontal length of upper glass _____ mm

rtical length of upper glass_____ mm

rizontal length of lower glass_____ the same as F1

ertical length of lower glass_____ mm

s generally longer than F2 when terminals are with pin.

***: Thickness of glass_____ mm

ndard type: 1.1 mm or 0.7 mm

ickness of LCD_____ mm

al: ☐Right ☐Left ☐Right or Left

V1 :Horizontal length of viewing area _____ mm

V2 :Vertical length of viewing area _____ mm

CN**: Terminal length_____ mm

CS**:Terminal length_____ mm

**CN or CS=0 in case of one side terminal type.

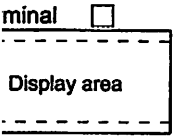
CC: Terminal length_____ mm

SE,SW,SN,SS : Seal width

(According to design or manufacturing condition:
about 2.0 mm to 4.0 mm)

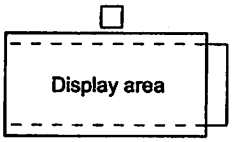
iel Form

minal ☐

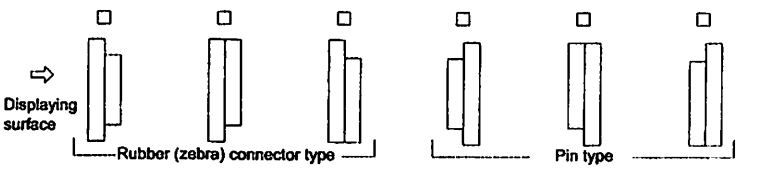


- Type A -

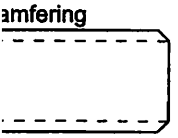
☐



- Type B -

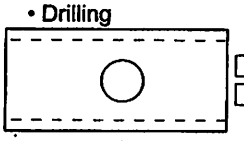


amfering



☐ Yes ☐ No

• Drilling



☐ Yes ☐ No

plays Mode

ing angle : ☐6 o'clock ☐ 12 o'clock ☐ _____ o'clock

: ☐TN ☐FSTN (Black and white)

STN: (☐ Yellow green ☐ Gray ☐ Blue)

hromaticity coordinates (____≦ x ≦ ____ , ____≦ y ≦ ____)

Positive type ☐Negative type

Reflective ☐Transflective ☐Transmissive

rential specifications:

Response time t_{on} ____ ms (____ °C) t_{off} ____ ms (____ °C)

Viewing angle ____ deg. (____ °C) ☐Contrast ____ (____ °C)

Others_____

arizer

ce finishing: ☐Normal ☐Anti-glare ☐_____

: ☐Normal (neutral gray) ☐Red ☐Green

☐Blue ☐_____

polarizer : ☐ Attached type ☐ Separate type

polarizer : ☐ Attached type ☐ Separate type

ing Method

tic ☐ Multiplexing: (1/_____ duty, 1/_____ bias)

ating voltage (V_{opr}) :_____ V

e frequency:_____ Hz

g IC:_____ (Manufacturer _____)

nt consumption:_____ μA

10. Temperature Range

Operating temperature range

☐With temperature compensation circuit (or volume)

(☐ 0°C to 50°C ☐ _____ °C to _____ °C)

☐Without temperature compensation circuit

(☐ 0°C to 50°C ☐ _____ °C to _____ °C)

Storage temperature range

(☐ - 20°C to 60°C ☐ _____ °C to _____ °C)

11. Terminal Connecting Method

☐Rubber connector (Zebra rubber)

☐Pin: ☐DIL ☐SIL ☐_____

Pitch (☐2.54 ☐ _____ mm) Length (_____ mm)

☐Heat seal: ☐Equipped ☐Unnecessary

12. Others

Print (Characters, lines, masks etc.) : ☐Yes ☐No

Protective film:

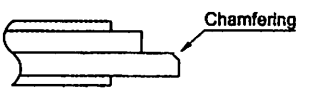
☐Yes (Color: ☐ Red ☐Translucent ☐Transparent) ☐No

Chamfering (for heat-seal connector) :

☐Yes (Position:_____)

(Quantity:_____)

☐No



13. Schedule

Estimate :_____

Sample : Delivery_____ , Quantity :_____ pcs

Mass production : Target price : _____

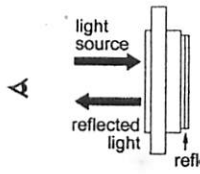
Delivery _____ , Total quantity:_____ pcs

Quantity per month:_____ pcs

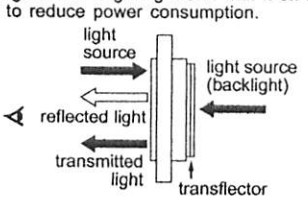
uid Crystal Display Modules

FLECTIVE/TRANSFLECTIVE/TRANSMISSIVE LCD

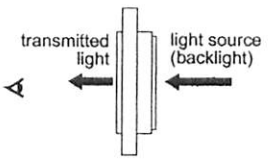
eflective LCD
ector bonded to the rear polarizer
cts the incoming ambient light. Low power
sumption because no backlight is required.



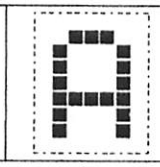
❷ Transflective LCD
Transflector bonded to the rear polarizer reflects light from the front as well as enabling lights to pass through the back. Used with backlight off in bright light and with it on in low light to reduce power consumption.



❸ Transmissive LCD
Without reflector or transflector bonded to the rear polarizer. Backlight required. Most common is transmissive negative image.



SITIVE/NEGATIVE MODE

Positive type	

Negative type		Negative type (inverse image) (when data is inverted)	

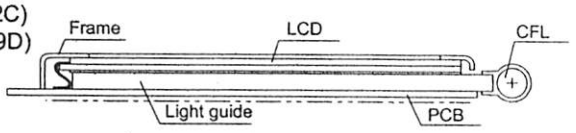
TYPE/STN TYPE/FSTN TYPE

	(Background/dot color) Gray/Black	TN(Twisted Nematic) type is most conventional and economical. It is used for static drive LCD and low-duty drive LCD (watch,calculator, etc.)
	Yellowgreen/Dark blue Gray/Dark blue White/Blue	STN (Super Twisted Nematic) type has a higher twist angle, and thus provides clear visibility and wider viewing angle. This is suitable especially for high-duty drive LCD.
N	White/Black	FSTN (Film Super Twisted Nematic) type utilizes RCF (Retardation Control Film) to remove the coloring of STN LCD. Thus FSTN type provides easy-to-read black-and-white display.

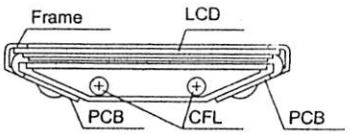
RUCTURE AND FEATURE OF LCD MODULE WITH BACKLIGHT

L (Cold Cathode Fluorescent Lamp) backlight
atures: high brightness, long service life, inverter required

dge backlight type
(G2446,G242C)
(G321D,G649D)

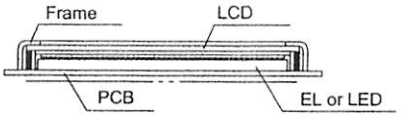


acklight type



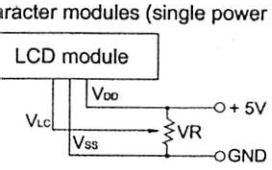
EL (Electroluminescent Lamp) backlight
LED (Light Emitting Diode) backlight

Features: EL: thin, inverter required
LED: long service life, low voltage driving, no inverter required

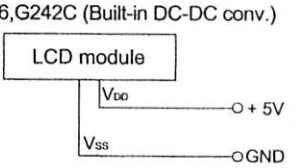


WER SUPPLY

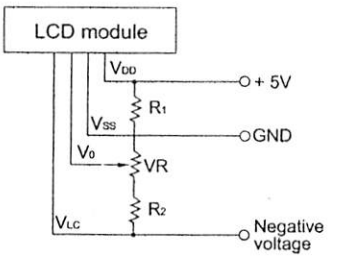
aracter modules (single power supply)



• G2446,G242C (Built-in DC-DC conv.)

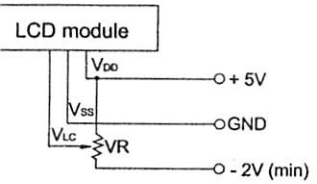


• G321D, G324E and G649D

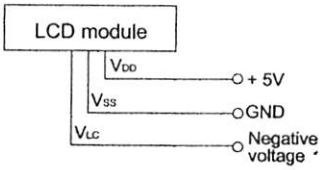


Note 1:Contrast can be adjusted by VR.
Note 2:For module with backlight, power supply for backlight is necessary.

aracter Modules(Dual power supply)



• Y1206 and G1226



·Negative voltage should be variable for contrast adjustment.

Special thanks to:

- Allah SWT, yang dengan segala rahmat, bimbingan, dan lindungan-Nya dan dengan segala kuasa-Nya memberiku hidayah sehingga aku dapat menyelesaikan semua yang tertunda.
- Bapak Ir. Sidik Noertjahjono, MT selaku dekan FTI.
- Bapak Ir. F. Yudi Limpraptomo, MT, selaku dosen pembimbing. Terima kasih atas kesempatannya sehingga saya dapat menyelesaikan skripsi saya yang telah lama tertunda, sehingga akhirnya saya bisa lulus dan mendapatkan gelar ST.
- Ibu Ir. TH. Mimin Mustikawati, MT dan Bapak I Komang Somawirata, ST, MT, selaku dosen penguji. Terima kasih atas segala bimbingan dan arahannya.
- Istriku tercinta, Fitria Yanti, yang selalu memberi semangat padaku dan tidak henti2nya mengomeliku.
- My little princess, Safira Putri Febrianti, yang tangisannya setia menemani malam2 perjuanganku.
- Bapak dan Mama di Pandaan, yang tidak henti-hentinya memberi doa dan dorongan untukku.
- Papa dan Ibu di Lawang, yang memberiku kesempatan untuk menyelesaikan tugasku.
- Vica, Disti, Irwan dan Angga, adik2ku yang membantuku dengan doa walaupun nun jauh di Lawang dan Surabaya.
- Mas Agung, yang telah menemaniku belajar.
- Blacky : *you're the best.* Yang telah mengantarku ujian.

- My Mio yang setia selalu mengantarku ke kampus.
- Teman2 seperjuanganku yang ola olo : Santo Rukmono & Doni Rohmadi
- Dan semua keluarga dan sahabat yang tidak bisa aku tulis karena terlalu banyaknya. Tapi aku mengucapkan terima kasih atas semua dukungannya.