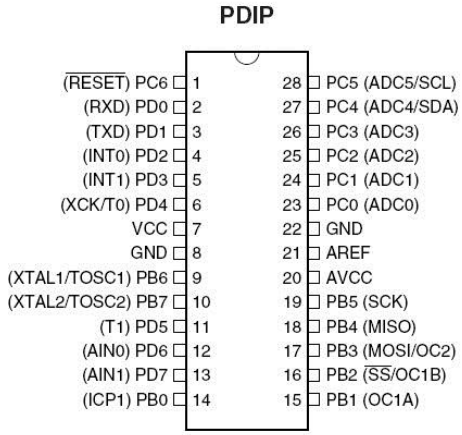
BAB III

LANDASAN TEORI

3.1 *Microcontroller* Atmega 8

AVR (*Advanced Versatile RISC*) merupakan salah satu jenis *microcontroller* yang di dalamnya terdapat berbagai macam fungsi. Perbedaannya pada mikro yang pada umumnya digunakan seperti MCS51 adalah pada AVR tidak perlu menggunakan *oscillator* eksternal karena di dalamnya sudah terdapat internal *oscillator*. Selain itu kelebihan dari AVR adalah memiliki *Power-On Reset*, yaitu tidak perlu ada tombol *reset* dari luar karena cukup hanya dengan mematikan *supply*, maka secara otomatis AVR akan melakukan *reset*. Untuk beberapa jenis AVR terdapat beberapa fungsi khusus seperti ADC, EEPROM sekitar 128 byte sampai dengan 512 byte[2].

AVR Atmega 8 adalah *microcontroller* CMOS 8-*Bit* berarsitektur AVR RISC yang memiliki 8 Kbyte *in-System Programmable Flash*. *Microcontroller* dengan konsumsi daya yang rendah ini mampu mengeksekusi instruksi dengan kecepatan maksimum 16 MIPS pada frekuensi 16 MHz. Jika dibandingkan dengan Atmega 8L perbedaannya hanya terletak pada besarnya tegangan yang diperlukan untuk bekerja. Untuk Atmega 8 tipe L, *microcontroller* ini dapat bekerja dengan tegangan antara 2,7-5,5 V sedangkan untuk Atmega 8 hanya dapat bekerja pada tegangan antara 4,5–5,5 V[2].



Gambar 3.1 Susunan Pin *Microcontroller* Atmega 8

Atmega 8 memiliki 28 Pin, yang masing-masing pinnya memiliki fungsi yang berbeda-beda baik sebagai *Port* maupun fungsi yang lainnya. Berikut akan dijelaskan fungsi dari masing-masing kaki Atmega 8[2].

a. VCC

Merupakan *supply* tegangan +5V.

b. GND

Merupakan *supply* tegangan 0V.

c. *Port* B (PB7...PB0)

Didalam *Port* B terdapat XTAL1, XTAL2, TOSC1, TOSC2. Jumlah *Port* B adalah 8 buah pin, mulai dari pin B.0 sampai dengan B.7. Tiap pin dapat digunakan sebagai *input* maupun *output*. *Port* B merupakan sebuah 8-*Bit* bi-*directional* I/O dengan *internal pull-up* resistor. Sebagai *input*, pin-pin yang terdapat pada *Port* B *external pull-down*, akan mengeluarkan tegangan jika *pull-up* resistor diaktifkan. Khusus PB6 dapat digunakan sebagai *input* kristal (*inverting oscillator amplifier*) dan *input* ke rangkaian *clock* internal, tergantung

pada pengaturan *Fuses Bit* digunakan untuk memilih sumber *clock*. Sedangkan untuk PB7 dapat digunakan sebagai *output* kristal (*output oscillator amplifier*) tergantung pada pengaturan *Fuses Bit* yang digunakan untuk memilih sumber *clock*. Jika sumber *clock* yang dipilih dari *oscillator* internal (*Syncronous*), PB7 dan PB6 dapat digunakan sebagai I/O. Apabila menggunakan *oscillator* eksternal (*Asyncronous Timer*/*Counter*2) maka PB6 dan PB7 (TOSC2 dan TOSC1) digunakan untuk saluran *input timer*.

d. *Port* C (PC5…PC0)

Port C merupakan sebuah 7-*Bit* bi-*directional* I/O *Port* yang di dalam masing-masing pin terdapat *pull-up* resistor. Jumlah pinnya hanya 7 buah mulai dari pin C.0 sampai dengan pin C.6. Sebagai keluaran/*output*, *Port* C memiliki karakteristik yang sama dalam hal menyerap arus (*sink*) ataupun mengeluarkan arus (*source*).

e. RESET/PC6

Jika RSTDISBL *Fuses* diprogram, maka PC6 akan berfungsi sebagai pin I/O. Pin ini memiliki karakteristik yang berbeda dengan pin-pin yang terdapat pada *Port* C lainnya. Namun jika RSTDISBL *Fuses* tidak diprogram, maka pin ini akan berfungsi sebagai *input reset*. Dan jika *level* tegangan yang masuk ke pin ini rendah (0V) dan pulsa yang ada lebih pendek dari pulsa minimum (1.5 us), maka akan menghasilkan suatu kondisi *reset* meskipun *clock*-nya tidak bekerja.

f. *Port* D (PD7…PD0)

Port D merupakan 8-*Bit* bi-*directional* I/O dengan internal *pull-up* resistor. Fungsi dari *Port* ini sama dengan *Port-Port* yang lain. Hanya saja pada

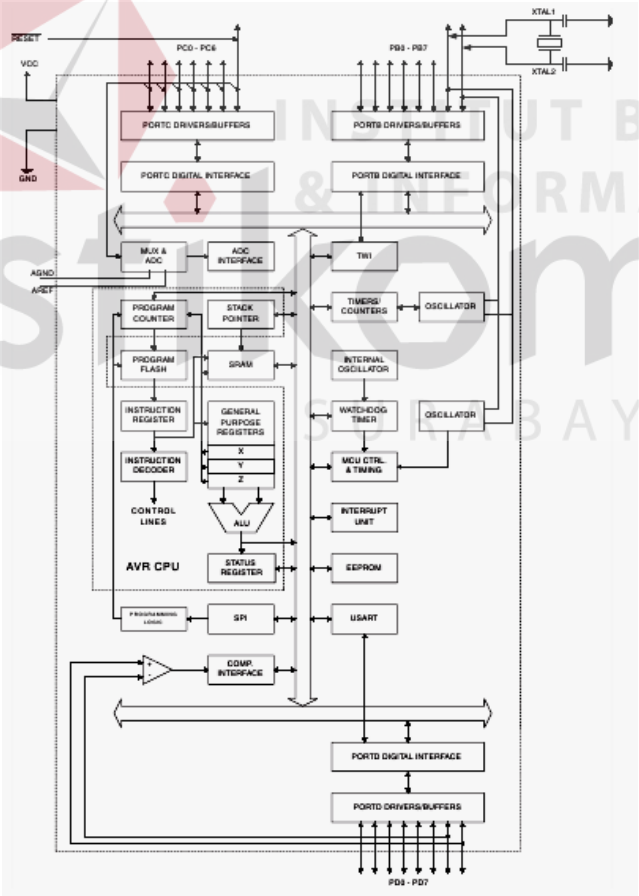
Port ini tidak terdapat kegunaan-kegunaan yang lain. Pada *Port* ini hanya berfungsi sebagai masukan dan keluaran saja atau biasa disebut dengan I/O.

g. AVcc

Pin ini berfungsi sebagai *supply* tegangan untuk ADC. Pin ini harus dihubungkan secara terpisah dengan VCC karena pin ini digunakan untuk analog saja. Bahkan jika ADC pada AVR tidak digunakan tetap saja disarankan untuk menghubungkannya secara terpisah dengan VCC. Jika ADC digunakan, maka AVcc harus dihubungkan ke VCC melalui *low pass filter*.

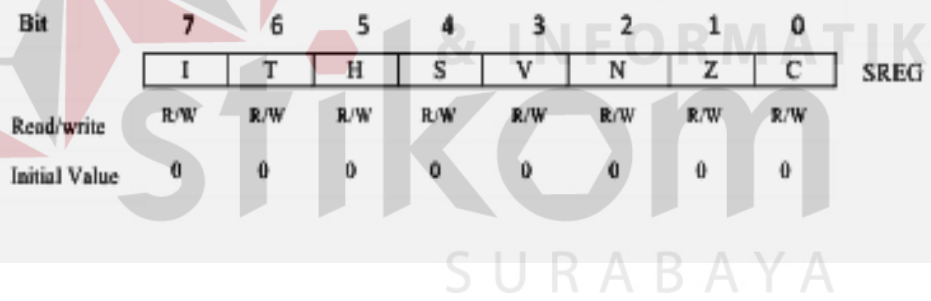
h. AREF

Merupakan pin referensi jika menggunakan ADC.



Gambar 3.2 Blok Diagram *Microcontroller* Atmega 8

Status *Register* berisi informasi tentang hasil dari instruksi aritmatika yang terakhir dieksekusi. Informasi ini dapat digunakan untuk mengubah aliran program untuk melakukan operasi kondisional. *Register* ini di-*update* setelah semua operasi ALU (*Arithmetic Logic Unit*) dilakukan, seperti ditentukan dalam *Instruction Set Reference*. Hal ini mengurangi penggunaan instruksi perbandingan, sehingga prosesnya lebih cepat dan lebih terstruktur. Status *Register* ini tidak secara otomatis tersimpan ketika menjalankan interupsi dan juga ketika menjalankan sebuah perintah setelah kembali dari interupsi. Sehingga hal tersebut harus dilakukan melalui *software*. Status *register* ada pada Gambar 3.3 berikut ini.



Gambar 3.3 Status *Register* Atmega 8

i. *Bit* 7(I)

Merupakan *bit Global Interrupt Enable*. *Bit* ini harus di-*set* agar semua perintah interupsi dapat dijalankan. Jika *bit* ini di-reset, maka semua perintah interupsi baik yang individual maupun yang secara umum akan di abaikan. *Bit* ini akan di-*reset* oleh *software* setelah sebuah interupsi dijalankan dan akan di-*set*

kembali oleh perintah RETI (*Return from interrupt*). *Bit* ini juga dapat di-*set* dan di-*reset* melalui aplikasi dan intruksi SEI (*Set Enable Interrupt*) dan CLI (*Clear Interrupt*).

j. *Bit* 6(T)

Merupakan *bit Copy Storage*. Instruksi *Bit Copy* BLD (*Bit LoaD*) dan BST (*Bit STorage*) menggunakan T-*bit* sebagai sumber atau tujuan untuk *bit* yang telah dioperasikan. Sebuah *bit* dari sebuah *register* dalam *Register File* dapat disalin ke dalam T-*bit* dengan menggunakan instruksi BST, dan sebuah *bit* di dalam T-*bit* ini dapat disalin ke dalam *bit* di dalam *register* pada *Register File* dengan menggunakan perintah BLD.

k. *Bit* 5(H)

Merupakan *bit Half Carry Flag*. *Bit* ini menandakan sebuah *Half Carry* dalam beberapa operasi aritmatika. *Bit* ini berfungsi dalam aritmatika BCD.

l. *Bit* 4(S)

Merupakan *Sign Bit*. *Bit* ini merupakan *bit* ekslusif atau di antara *Negative Flag* (N) dan *two's Complement Overflow Flag* (V).

m. *Bit* 3(V)

Merupakan *Bit Two's Complement Overflow Flag*. *Bit* ini menyediakan fungsi aritmatika dua komplemen.

n. *Bit* 2(N)

Merupakan *Bit Negative Flag*. *Bit* ini mengindikasikan sebuah hasil *negative* di dalam sebuah fungsi logika atau aritmatika.

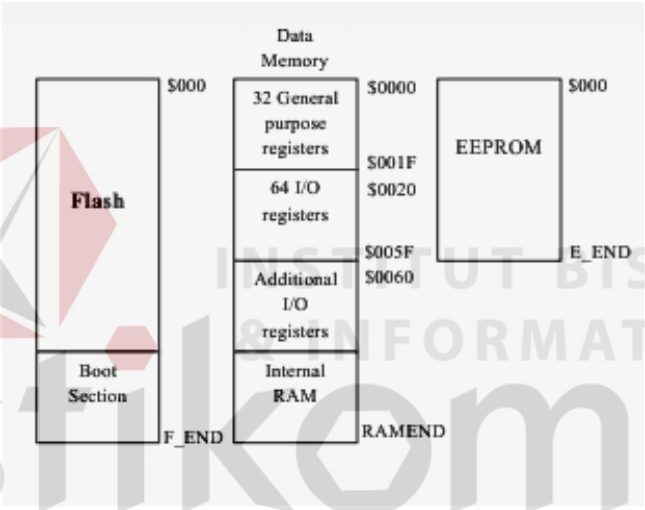
o. *Bit* 1(Z)

Merupakan *Bit Zero Flag*. *Bit* ini mengindikasikan sebuah hasil nol "0" dalam sebuah fungsi aritmatika atau logika.

p. *Bit* 0(C)

Merupakan *Bit Carry Flag*. *Bit* ini mengindikasikan sebuah *Carry* atau sisa dalam sebuah aritmatika atau logika.

3.1.1 Memori AVR Atmega



Gambar 3.4 Peta Memori Atmega

Memori Atmega terbagi menjadi tiga yaitu[2] :

1. Memori *Flash*

Memori *flash* adalah memori ROM tempat kode-kode program berada. Kata *flash* menunjukan jenis ROM yang dapat ditulis dan dihapus secara elektrik. Memori *flash* terbagi menjadi dua bagian yaitu bagian aplikasi dan bagian *boot*. Bagian aplikasi adalah bagian kode-kode program apikasi

berada. Bagian *boot* adalah bagian yang digunakan khusus untuk *boot*ing awal yang dapat diprogram untuk menulis bagian aplikasi tanpa melalui *programmer/downloader*, misalnya melalui USART (*Universal Serial Asynchronous Receiver Transmitter*).

2. Memori Data

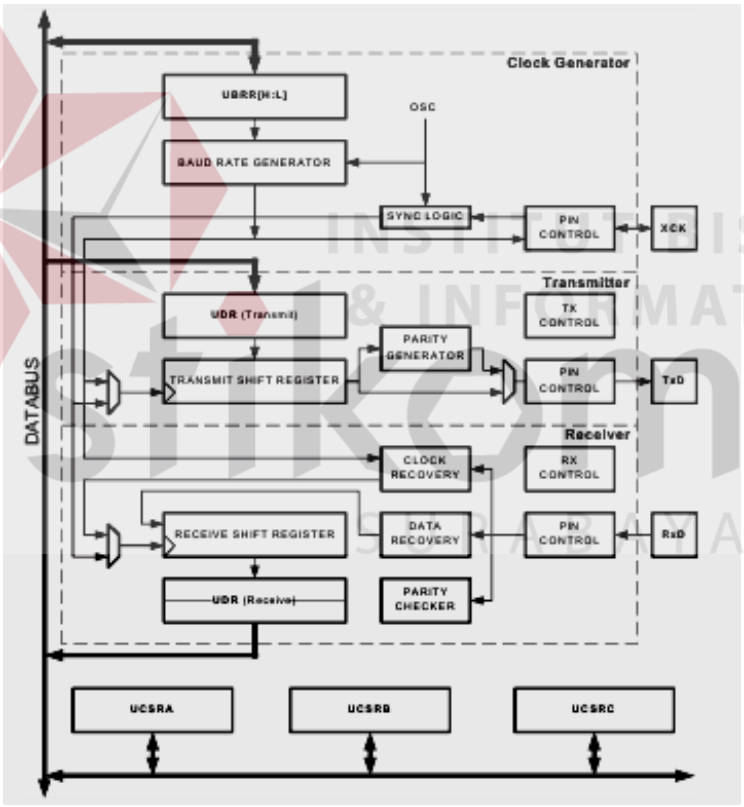
Memori data adalah memori RAM yang digunakan untuk keperluan program. Memori data terbagi menjadi empat bagian yaitu, 32 GPR (*General Purpose Register*), 64 I/O *register, Additional* I/O, dan 1024 internal RAM. GPR adalah *register* khusus yang bertugas untuk membantu eksekusi program oleh ALU (*Arithmatich Logic Unit*), dalam instruksi *assembler* setiap instruksi harus melibatkan GPR. Dalam bahasa C biasanya digunakan untuk *variabel* global atau nilai balik fungsi dan nilai-nilai yang dapat memperingan kerja ALU. Dalam istilah *processor* komputer sehari-hari GPR dikenal sebagai "*cache memory*". I/O *register* dan *Aditional* I/O *register* adalah *register* yang difungsikan khusus untuk mengendalikan berbagai *pheripheral* dalam *microcontroller* seperti pin *Port*, *timer/counter*, USART dan lain-lain. *Register* ini dalam keluarga *microcontroller* MCS51 dikenal sebagi SFR (*Special Function Register*).

3. EEPROM

EEPROM adalah memori data yang masih dapat menyimpan data walaupun ketika *chip microcontroller* dalam keadaan mati (*off*), digunakan untuk keperluan penyimpanan data yang tahan terhadap gangguan catu daya.

3.1.2 Komunikasi Serial Pada Atmega 8

Microcontroller AVR Atmega 8 memiliki *Port* USART pada pin 2 dan pin 3 untuk melakukan komunikasi data antara *microcontroller* dengan *microcontroller* ataupun *microcontroller* dengan komputer. USART dapat difungsikan sebagai transmisi data sinkron dan asinkron. Sinkron berarti *clock* yang digunakan antara *transmitter* dan *receiver* satu sumber *clock*. Sedangkan asinkron berarti *transmitter* dan *receiver* mempunyai sumber *clock* sendiri-sendiri. USART terdiri dalam tiga blok yaitu *clock generator*, *transmitter*, dan *receiver*[2].



Gambar 3.5 Blok USART

3.1.3 *Clock Generator*

Clock generator berhubungan dengan kecepatan transfer data (*baud rate*), *register* yang bertugas menentukan *baud rate* adalah *register* UBRR (USART *Baud Rate Register*) [2].

Tabel 3.1 *Baud Rate* Atmega 8

Operating Mode	*Equation for Calculating Baud Rate*	*Equation for Calculating* UBRR *Value*
Asyncrhronous Normal Mode (U2X = 0)	$\text{Baud} = \frac{fosc}{16(\text{UBRR}+1)}$	$\text{UBRR} = \frac{fosc}{16(\text{BAUD}+1)} - 1$
Asyncrhronous Double Speed Mode (U2X = 1)	$\text{Baud} = \frac{fosc}{8(\text{UBRR}+1)}$	$\text{UBRR} = \frac{fosc}{8(\text{BAUD}+1)} - 1$
Syncrhronous Master Mode	$\text{Baud} = \frac{fosc}{2(\text{UBRR}+1)}$	$\text{UBRR} = \frac{fosc}{2(\text{BAUD}+1)} - 1$

Dimana :

1. Fosc adalah frekuensi *oscilator* yang digunakan.
2. BAUD adalah transfer *Bit* per detik.

3.1.4 USART *Transmitter*

USART *transmitter* berhubungan dengan data pada Pin TX. Perangkat yang sering digunakan adalah *register* UDR sebagai tempat penampungan data yang akan ditransmisikan. *Flag* TXC sebagai indikator bahwa data yang ditransmisikan telah sukses (*complete*), dan *flag* UDRE sebagai indikator jika UDR kosong dan siap untuk diisi data yang akan ditransmisikan lagi[2].

3.1.5 USART *Receiver*

USART *receiver* berhubungan dengan penerimaan data dari Pin RX. Perangkat yang sering digunakan adalah *register* UDR sebagai tempat penampung data yang telah diterima, dan *flag* RXC sebagai indikator bahwa data telah sukses (*complete*) diterima[2].

3.2 *Global Positioning System*(GPS)

GPS (*Global Positioning System*) adalah sebuah sistem navigasi berbasis satelit yang dibangun dari sebuah jaringan yang terdiri dari 24 satelit yang diletakkan dalam orbit. GPS bekerja pada berbagai kondisi cuaca, di manapun posisi di dunia, dan 24 jam satu hari[1].

GPS adalah sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga dimensi, serta informasi mengenai waktu. GPS terdiri dari 3 segmen yaitu segmen angkasa, kontrol/pengendali, dan pengguna. Segmen angkasa terdiri dari 24 satelit yang beroperasi dalam 6 orbit pada ketinggian 20.200 km dengan periode 12 jam (satelit akan kembali ke titik yang sama dalam 12 jam). Segmen Kontrol/Pengendali mempunyai pusat pengendali utama yang terdapat di Colorodo Springs, dan 5 stasiun pemantau lainnya, serta 3 antena yang tersebar di bumi ini. Pada sisi pengguna dibutuhkan penerima GPS yang biasanya terdiri dari penerima, prosesor, dan antenna[3].

Sebuah GPS *receiver* harus terkunci dengan sinyal dari setidaknya 3 satelit untuk menghitung posisi 2D (*latitude* dan *longitude*) dan pergerakan *track*. Dengan 4 atau lebih satelit, *receiver* dapat menentukan posisi 3D *user* (*latitude*, *longitude*, dan *altitude*). Begitu posisi *user* sudah ditentukan, unit GPS dapat menghitung informasi lain, seperti kecepatan, pelacakan, jarak tempuh, jarak tujuan, waktu matahari terbit dan terbenam, dan sebagainya[1].

3.2.1 Protokol NMEA 0183

Protokol NMEA 0183 (*National Marine Electronics Association*) merupakan suatu badan yang menerbitkan spesifikasi dan mendeskripsikan berbagai perlengkapan navigasi agar dapat berkomunikasi satu sama lain melalui koneksi serial RS-232 atau lainnya (misalnya USB *Port*). NMEA menggunakan file data ASCII dalam mentransmisikan sistem informasi GPS dari *receiver* ke *hardware* yang berfungsi sebagai *input* dari posisi secara *realtime* untuk navigasi. Salah satu aplikasi protokol ini adalah pada komunikasi data GPS[4].

Parameter yang digunakan oleh protokol ini adalah sebagai berikut[4] :

Baudrate : 4800

Jumlah data : 8 *Bit*

Stop Bit : 1

Parity : *None*

3.2.2 Format Data GPS

Data GPS terdiri dari kalimat teks yang mengandung lintang, bujur, dan informasi lainnya. Setiap kalimat terdiri dari awalan, ditambah satu atau lebih blok data, masing-masing dipisahkan oleh koma[4].

GPS menerima data dari satelit dan mengirimkannya ke bagian *output* dengan format data yang beragam. Data yang dikirimkan oleh GPS mengacu pada standar NMEA 0183, yaitu standar kalimat laporan yang dikeluarkan oleh GPS *receiver*. Standar NMEA memiliki banyak jenis bentuk kalimat laporan diantaranya koordinat lintang (*latitude*), bujur (*longitude*), ketinggian (*altitude*), waktu sekarang standar UTC (UTC *Time*) dan kecepatan (*speed over ground*) [4].

Setiap data diawali dengan karakter $ dan diakhiri dengan <CR><LF>. Pada prakteknya tidak semua data dengan *header* ini diambil, hanya yang menyangkut waktu, garis lintang dan garis bujur untuk posisi pengguna[4].

Tabel 3.2 Format Data GPS

Tipe String	Deskripsi
$GPAAM	*Waypoint Arrival Alarm*
$GPALM	*GPS Almanac Data*
$GPBEC	*Bearing & Distance to Waypoint, Dead Reckoning*
$GPBOD	*Bearing, Origin to Destination*
$GPBWC	*Bearing & Distance to Waypoint, Great Circle*
$GPFSI	*Frequency Set Information*
$GPGGA	**Global Positioning System Fix Data (Time, Position, Elevation)*
$GPGLC	*Geographic Position, Loran-C*
$GPGLL	**Geographic Position, Latitude/Longitude*
$GPGRS	*GPS Range Residuals*
$GPGSA	**GPS DoP (Dilution of Precision) and Active Satellites*
$GPGSV	**GPS Satellites in View*
$GPHDG	*Heading, Deviation & Variation*
$GPHDT	*Heading, True*
$GPHSC	*Heading Steering Command*
$GPMWV	*Wind Speed and Angle*
$GPRMC	**Recommended Minimum Specific GNSS GPS/TRANSIT Data (Time, Position, Velocity)*
$GPROT	*Rate of Turn*
$GPRPM	*Revolutions*
$GPRTE	*Routes*

Berikut ini adalah penjelasan mengenai beberapa kalimat yang sering digunakan dalam format NMEA :

1. RMB (*Recommended Minimum Navigation Information*)

RMB adalah kalimat "rekomendasi navigasi minimum", dan dikirim setiap kali sebuah rute atau sebuah *goto* (seperti misalnya menetapkan tujuan atau *waypoint*) diaktifkan.

Beberapa sistem di-*set* untuk mengirimkan kalimat ini sepanjang waktu, mengirimkan data *null* jika tidak ada *goto* yang dipilih, sementara pada sistem lain hanya mengirimkannya saat diperlukan. Format kalimatnya adalah :

```
$GPRMB,A,x.x,a,c--c,d—
d,llll.ll,e,yyyyy.yy,f,g.g,h.h,i.i,j*kk
```

Di mana :

RMB : Informasi rekomendasi navigasi minimum

A : Status data (A = *Active*, V = *Void*)

x.x : *Cross-track error* (diukur dalam mil laut, nilai maksimumnya 9.99)

a : Arah kemudi untuk memperbaiki *cross-track error* (L = *Left*, R = *Right*)

c--c : Nomor identitas *waypoint* asli

d--d : Nomor identitas *waypoint* tujuan

llll.ll : Garis lintang (*latitude*) *waypoint* tujuan

e : N (*North*) atau S (*South*) untuk *latitude*

yyyyy.yy : Garis bujur (*longitude*) *waypoint* tujuan

f : E (*East*) atau W (*West*) untuk *longitude*

g.g : Jarak ke tujuan (diukur dalam mil laut, nilai maksimumnya 999.9)

h.h : Arah untuk sampai ke tujuan, dalam derajat

j : Status kedatangan (A = *arrived*, V = *not arrived*)

2. RMC (*Recommended Minimum Specific* GPS/Transit Data)

Kalimat RMC merupakan data ekuivalen NMEA untuk PVT (*Position, Velocity, Time*) yaitu Posisi, Kecepatan, Waktu. Format kalimatnya adalah:

```
$GPRMC,hhmmss.ss,A,llll.ll,e,yyyyy.yy,f,x.x,y.y,ddmm
yy,z.
z,a*hh
```

Di mana :

RMC : *Recommended Minimum Sentence* C

hhmmss.ss :Waktu saat pemosisian, dalam UTC (*Universal Time Coordinated*)

A : Status data (A = *Active*, V = *Void*)

llll.ll : Garis lintang (*latitude*) *waypoint* tujuan

e : N (*North*) atau S (*South*) untuk *latitude*

yyyyy.yy : Garis bujur (*longitude*) *waypoint* tujuan

f : E (*East*) atau W (*West*) untuk *longitude*

x.x : Kecepatan terhadap tanah pantauan (dalam mil laut)

y.y : Sudut penjejakan yang dipantau (dalam derajat)

ddmmyy : Tanggal UT (*Universal Time*)

z.z : Variasi magnetik

a : E (*East*) atau W (*West*) untuk variasi magnetik

*hh : *Checksum*

3. GGA (*Global Positioning System Fix Data*)

Kalimat GGA menyediakan lokasi 3 dimensi dan data keakuratan. Format kalimatnya adalah:

```
$GPGGA,hhmmss.ss,llll.ll,e,yyyyy.yy,f,a,bb,x.x,y.y,M
,z.z,
M,s.s,####*hh
```

Di mana :

GGA : *Global Positioning System Fix Data*

hhmmss.ss : Waktu saat pemosisian, dalam UTC (*Universal Time Coordinated*)

llll.ll : Garis lintang (*latitude*) *waypoint* tujuan

e : N (*North*) atau S (*South*) untuk *latitude*

yyyyy.yy : Garis bujur (*longitude*) *waypoint* tujuan

f : E (*East*) atau W (*West*) untuk *longitude*

1 : Kualitas pemosisian GPS

0 = *invalid*

1 = Pemosisian GPS (SPS)

2 = Pemosisian DGPS

3 = Pemosisian PPS (*Precise Positioning Service*)

4 = RTK (*Real Time Kinematic*)

5 = *Float* RTK

6 = Perkiraan (*deadreckoning*/perhitungan mati)

7 = *Input* secara manual

8 = Mode simulasi

bb : Jumlah satelit yang digunakan untuk menjejaki

x.x : Kesalahan *horizontal* (dilusi atau presisi)

y.y,M : Ketinggian antena (dalam meter)

z.z,M : Ketinggian geoid (permukaan air laut rata-rata), dalam meter

s.s : Waktu (dalam detik) sejak *update* terakhir

: Identitas stasiun DGPS

*hh : *Checksum*

4. VTG (*Actual Track Made Good and Speed Over Ground*)

Kalimat ini menyediakan informasi kecepatan terpantau

```
$GPVTG,t,T,?,??,s.ss,N,S.SS,K*hh
```

Di mana :

VTG : *Track* kecepatan di darat

t : Penjejakan sebenarnya

T : Teks yang sudah pasti (tetap) ini mengindikasikan bahwa penjejakan yang dipantau relatif terhadap utara

? : tidak digunakan

?? : tidak digunakan

s.ss : Kecepatan diatas permukaan tanah (dalam mil laut)

N : Teks yang sudah pasti (tetap) ini mengindikasikan bahwa kecepatan terhadap tanah adalah dalam knot

S.SS : Kecepatan terhadap tanah (dihitung dalam kilometer per jam)

K : Teks yang sudah pasti (tetap) ini mengindikasikan bahwa kecepatan terhadap tanah adalah dalam kilometer per jam

*hh : *Checksum*

5. RMA (*Navigation Data from Present Position*)

Kalimat ini menyediakan data navigasi berdasarkan posisi sekarang, format kalimatnya adalah:

```
$GPRMA,A,llll.ll,e,yyyyy.yy,f,?,??,s.ss,ccc,zz.z,a*hh
```

Di mana :

RMA : Data navigasi posisi sekarang

A : Status data (A = *Active*, V = *Void*)

llll.ll : Garis lintang (*latitude*) *waypoint* tujuan

e : N (*North*) atau S (*South*) untuk *latitude*

yyyyy.yy : Garis bujur (*longitude*) *waypoint* tujuan

f : E (*East*) atau W (*West*) untuk *longitude*

? : tidak digunakan

?? : tidak digunakan

s.ss : Kecepatan diatas permukaan tanah (dalam mil laut)

ccc : Arah terhadap daratan

zz.z : Variasi magnetik (variasi ke Timur dikurangi dari arah sesungguhnya)

a : E (*East*) atau W (*West*) untuk variasi magnetik

*hh : *Checksum*

6. GSA (GPS DoP *and Active Satellites*)

Kalimat ini menyediakan informasi terinci pada pemosisian oleh satelit. Dalam kalimat ini juga terdapat jumlah satelit yang digunakan dalam solusi saat ini dan DoP (*Dilution of Precision* / Dilusi Keakuratan) yaitu indikasi efek geometri satelit terhadap keakuratan pemosisian. DoP tidak memiliki satuan pengukuran, namun semakin kecil nilainya semakin baik. Format kalimat GSA adalah:

```
$GPGSA,A,B,x1,x2,x3,x4,x5,x6,x6,x8,x9,x10,x11,x12,x,
y,z*h
h
```

Di mana :

GSA : DoP dan satelit aktif

A : Mode pemosisian

M= *Manual* (di mana alat penerima dipaksa untuk bekerja pada 2 dimensi atau 3 dimensi)

A = *Automatic* (otomatis)

3 : Mode pemosisian

1 = Pemosisian tidak dimungkinkan

2 = 2 dimensi

3 = 3 dimensi

x1-x12 : ID dari masing-masing satelit (SV) yang digunakan untuk pemosisian

x : Dilusi keakuratan posisi

y : Dilusi keakuratan *horizontal*

z : Dilusi keakuratan *vertikal*

*hh : *Checksum*

7. GSV (*Satellites in View Data*)

Kalimat NMEA ini adalah yang paling informatif dimana menunjukkan data tentang satelit bahwa satelit tersebut mungkin mampu mencari berdasarkan *viewing mask* dan *almanac* data. GSV juga menunjukkan kemampuan unit saat ini untuk menjejaki data ini. Sebuah kalimat GSV dapat menyediakan data hingga empat buah satelit, sehingga mungkin diperlukan tiga buah kalimat GSV untuk memperoleh informasi lengkap. Untuk semua kalimat satelit GSV tidak perlu tampil secara berurutan. Pada GSV juga terdapat informasi SNR (*Signal-to-Noise Ratio*)

yang merupakan indikator kekuatan sinyal. Menurut standar NMEA, *range* SNR adalah 0 sampai 99 dB, jangkauan yang biasa bekerja pada GPS adalah 25-35 dB. Format kalimat GSV adalah:

```
$GPGSV,A,B,C,D1,E1,Az1,SNR1,D2,E2,Az2,SNR2,D3,E3,Az3
,SNR3,D4,E4,Az4,SNR4*hh
```

Di mana :

GSV : *Satelite in View*

A :Jumlah kalimat yang dibutuhkan untuk menampung data semua SV yang sedang dilihat

B : Nomor kalimat

C : Jumlah total satelit yang dilihat

D1-D4 : Nomor PRN (*Pseudo Random Number*) satelit

E1-E4 : Sudut elevasi (dalam derajat, nilai maksimumnya 90)

Az1-Az4 : Sudut azimuth (dalam derajat dari utara, nilainya 000 sampai 359)

SNR1-SNR4 : Nilai SNR (*Signal-to Noise Ratio*), semakin besar nilainya semakin baik sinyalnya

*hh : *Checksum*

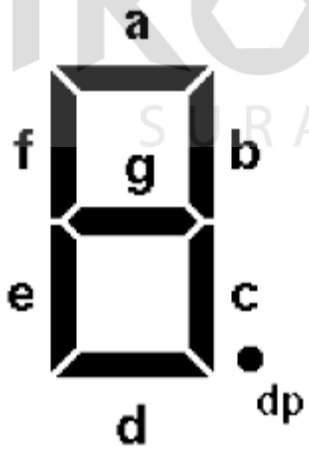
(Kathie Kingsley-Hughes, Hacking GPS, 2005, p199-p202) .

3.3 *Seven Segment*

Seven segment display adalah indikator penunjuk angka, terdiri dari tujuh buah LED (*Light emitting Diode*) yang disusun sehingga menjadi satu komponen. Dibawah ini Gambar 3.6 merupakan gambar *output* hasil tampilan *seven segment*:

Gambar 3.6 Tampilan *Seven Segment*

Setiap LED pada penampil *seven segment* diberi kode huruf untuk menyatakan LED mana yang nyala. Kode tersebut adalah a, b, c, d, e, f, g. sebagai contoh apabila yang menyala segmen a, b, g, e, dan d maka yang tampil adalah desimal 2. Berikut ini Gambar 3.7 adalah gambar posisi kode huruf pada penampil *seven segment*:

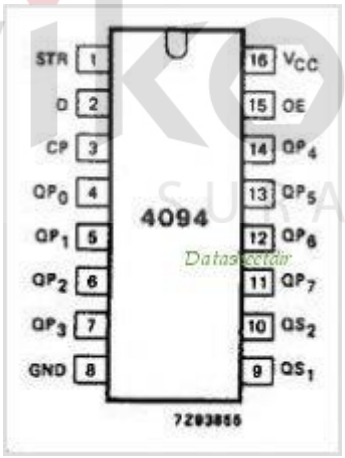


Gambar 3.7 Posisi Kode *Seven Segment Display*

Salah satu fungsi dari *seven segment* adalah untuk menampilkan sistem bilangan, penampil *seven segment* terdiri dari dua jenis yaitu *common anode* dan *common katode*. Pada *common anode* kaki-kaki anodanya terhubung ke *ground*, sebaliknya pada *common katode* kaki-kaki katodanya terhubung ke VCC.

3.4 IC *Shift Register* 4094

IC 4094 adalah IC *shift register* 8 bit yang memiliki *register latch* untuk setiap *bit* yang berguna untuk memindahkan data dari saluran *serial* kesaluran paralel dengan pergeseren *bit* Q0 sampai *bit* Q7 menuju *output*. *Output* paralel dapat dihubungkan langsung dengan jalur data umum. Data digeser ketika terjadi perubahan sinyal *clock* dari *Low* ke *High*, selanjutnya data digeser dari *register* geser ke *register* penyimpanan, kemudian dengan memberikan logika *high* pada pin OE akan menggeser data dari register penyimpanan menuju register *output*[5].



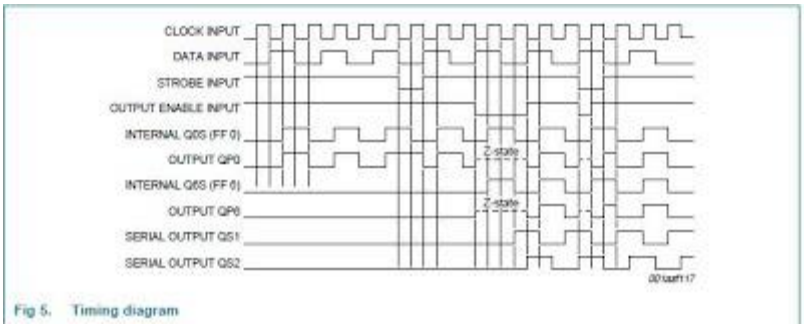
Gambar 3.8 IC *Shift Register* 4094

Tabel 3.3 Keterangan Pin IC 4094

No.	Nama Pin	Keterangan
1.	OE	*Output enable*
2.	QP0-QP7	*Output Paralel* 0 – *Output Paralel* 7
3.	D	*Input* Data Serial
4.	CP	*Clock Input*
5.	QS1-QS2	*Output* Serial1 - *Output* Serial2
6.	STR	*StrobeInput*
7.	VCC	V+
8.	GND	GND

3.4.1 Cara Kerja *Shift Register* 4094

Data masuk secara serial melalui pin D (1). Pada IC *Shift Register* ini data masuk baru disimpan setelah terjadi *clock*. Jadi cara memasukkan data pada *shift register* ini adalah *strobe* harus berlogika *high*. Kemudian data masuk-*clock*- data masuk-*clock*- data masuk-*clock*, begitu seterusnya. Setelah satu blok data dikirim, *strobe* harus dibuat berlogika *low* untuk mengakhiri pengiriman. Pin OE atau *Output Enable* digunakan untuk mengaktifkan *output* serial maupun *output* paralel. Logika 1 untuk *enable* dan logika 0 untuk *disable*. Q0 - Q7 adalah *output paralel* dari *shift register* ini sedangkan QS dan $\overline{QS}$ adalah *output* serial dari *shift register* ini. Jika menggunakan lebih dari satu IC *Shift Register* maka pin data dari IC *Shift Register* selanjutnya dihubungkan ke *output* serial dari IC *Shift Register* sebelumnya. Untuk lebih jelasnya bisa dilihat di *timing* diagram berikut ini[5]:



Gambar 3.9 *Timing* Diagram IC *Shift Register* 4094

3.5 Bahasa *BASIC*

Bahasa *BASIC* adalah salah satu bahasa tingkat tinggi (*High Level Language*) yang berorientasi ke pemecahan masalah (*problem solving*). *BASIC* yang merupakan singkatan dari *Beginner's All purpose Symbolic Instruction Code*, ditemukan oleh John G. Kemeny, profesor dari Darthmouth *College* dan Thomas E. Kurtz pada tahun 1960. Perintah-perintah dalam bahasa *BASIC* relatif mudah dipahami, baik oleh orang yang awam sekalipun[7].

Banyak sekali jenis *compiler* dari versi bahasa *BASIC* yang ada di pasaran, semisal : BASICA, GWBASIC, MBASIC, Turbo *BASIC*, *Quick BASIC*, Power *BASIC*, dll, akan tetapi pada dasarnya kesemuanya bermuara pada *style* pemrograman yang sama yaitu bahasa *BASIC* itu sendiri[7].

Bahasa *BASIC* kemudian dikembangkan dengan pemrograman yang lebih terstruktur, dengan tujuan agar sedapat mungkin dihindari penggunaan perintah *GOTO* yang menyebabkan program menjadi sukar dipahami alurnya. Pada pemrograman terstuktur terdapat perintah penyeleksian kondisi dan berbagai

macam alternatif perintah perulangan. Bahasa *BASIC* yang sudah terstruktur, semisal TURBO *BASIC* dan *Quick BASIC*[7].

Saat ini perkembangan bahasa BASIC sudah sedemikian pesatnya, sehingga terdapat *software* BASIC yang dapat dijalankan pada *platform Windows* dan pemrograman berorientasi obyek (*Object Oriented Programming*) seperti VISUAL *BASIC*[7].

3.6 BASCOM-AVR

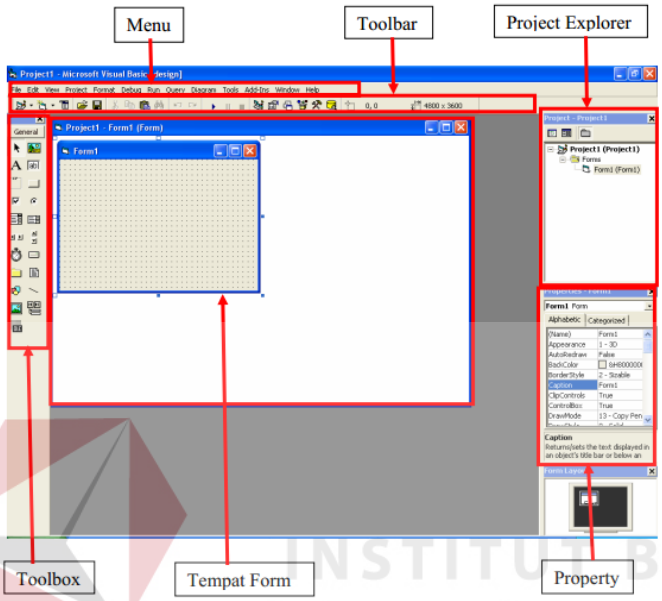
BASCOM-AVR adalah salah satu *software* pemrograman *microcontroller* yang menggunakan bahasa Basic. Selain pemahaman bahasanya yang mudah, *compiler* BASCOM-AVR juga dilengkapi dengan *simulator*. Sehingga mempermudah proses pemrograman[6].

3.7 Visual Basic 6.0

Visual *Basic* merupakan bahasa pemrograman yang sangat mudah dipelajari, dengan teknik pemrograman visual yang memungkinkan penggunanya untuk berkreasi lebih baik dalam menghasilkan suatu program aplikasi. Ini terlihat dari dasar pembuatan dalamVisual *Basic* adalah *form*, dimana pengguna dapat mengatur tampilan *form* kemudian dijalankan dalam *script* yang sangat mudah[8].

Ledakan pemakaian Visual *Basic* ditandai dengan kemampuan Visual *Basic* untuk dapat berinteraksi dengan aplikasi lain di dalam sistem operasi *Windows* dengan komponen *ActiveX Control*. Dengan komponen ini memungkinkan penguna untuk memanggil dan menggunakan semua model data

yang ada di dalam sistem operasi *Windows*. Hal ini juga ditunjang dengan teknik pemrograman di dalam Visual *Basic* yang mengadopsi dua macam jenis pemrograman yaitu Pemrograman Visual dan *Object Oriented Programming* (OOP) [8].



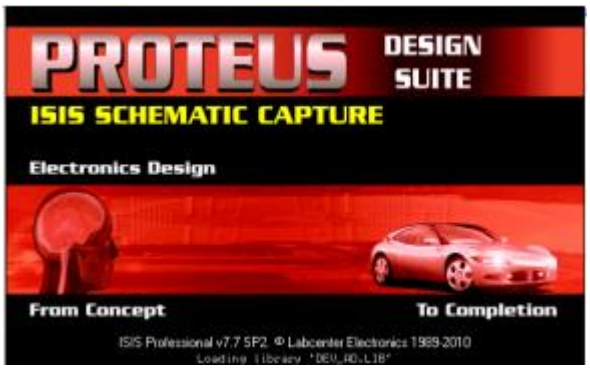
Gambar 3.10 *Interface* Visual *Basic* 6.0

Visual *Basic* 6.0 sebetulnya perkembangan dari versi sebelumnya dengan beberapa penambahan komponen yang sedang tren saat ini, seperti kemampuan pemrograman internet dengan DHTML (*Dynamic Hyper Text Mar Language*), dan beberapa penambahan fitur *database* dan multimedia yang semakin baik[8].

3.8 Proteus ISIS *Schematic Capture*

Proteus adalah sebuah *software* simulasi yang sekaligus untuk mendesain rangkaian dan PCB. Proteus mengkombinasikan program ISIS untuk membuat

skematik desain rangkaian dengan program ARES untuk membuat *layout* PCB dari skematik yang kita buat[9].



Gambar 3.11 Tampilan Awal Proteus

Fitur-fitur Proteus adalah :

a. Memiliki kemampuan untuk mensimulasikan hasil rancangan baik digital maupun analog maupun gabungan keduanya
b. Memiliki fitur yang cukup lengkap dan mudah dipelajari
c. Dapat mensimulasikan berbagai jenis *microcontroller* seperti AVR, PIC, 8051 series, dan *microcontroller* lainnya.
d. Memiliki model-model *peripheral* yang *interactive* seperti LED, tampilan LCD, RS232, dan berbagai jenis *library* lainnya,
e. Menyediakan *instrument-instrument virtual* seperti *voltmeter*, *ammeter*, *osccilloscope*, *logicanalyser*, dll,
f. Memiliki kemampuan menampilkan berbagi jenis analisis secara grafis seperti *transient*, frekuensi, *noise*, distorsi, AC dan DC, dll.
g. Menyediakan berbagai jenis komponen-komponen analog,

h. Bersifat *open architecture* sehingga kita bisa memasukkan program seperti C++ untuk keperluan simulasi,

i. Dapat digunakan untuk pembuatan PCB yang di-*update* secara langsung dari program ISIS ke program pembuat PCB-ARES