BAB II

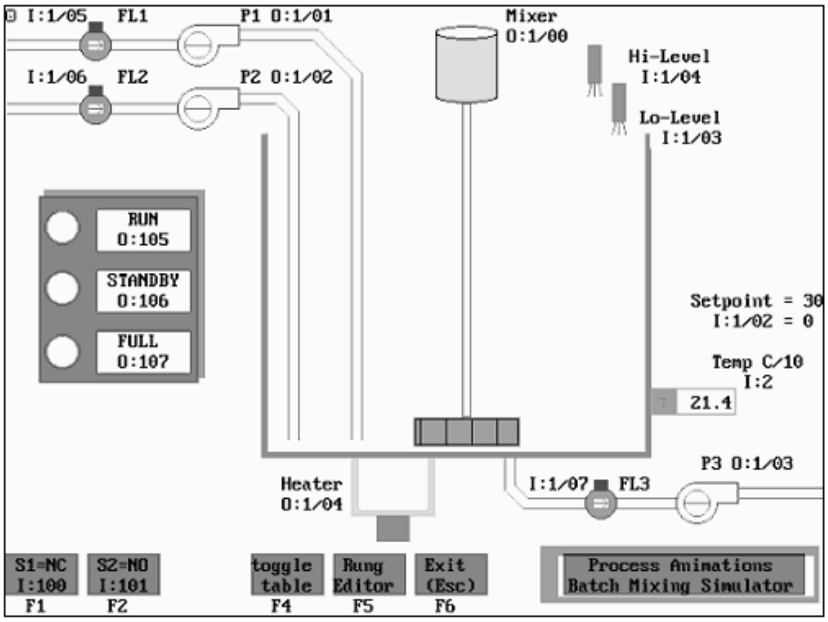
DASAR TEORI

2.1 Batch Mixer

Batch mixer merupakan alat yang menampung beberapa bahan baku untuk dicampurkan sehingga menjadi produk dapat digunakan. Bahan baku dapat berupa cairan, padat dan gas. Penggunaan bahan baku dan cara pencampuran bahan baku yang membedakan perancangan *batch mixer* pada setiap industri.

2.1.1 PSIM *Programmable Logic Control* (PLC) *Simulator*

Sekarang ini telah banyak *simulator* mengenai cara kerja batch mixer, salah satunya PSIM PLC *Simulator*. PSIM PLC *Simulator* adalah perangkat lunak yang berisi latihan simulasi PLC menggunakan jenis pemrograman PLC Allen Bradley. Dalam perangkat lunak ini terdiri dari simulasi proses dan simulasi aplikasi industri, yaitu simulasi I/O, simulasi *traffic light*, simulasi *conveyor*, dan simulasi *batch mixer*.



Gambar 2.1 *Batch Mixer* pada PSIM PLC *Simulator*

Berikut cara kerja *batch mixer* pada PSIM PLC *Simulator*.

1. Saat *batch mixer* dioperasikan, maka lampu indikator *RUN* menyala.
2. Pompa 1 dan pompa 2 mengisi bahan baku ke tangki pencampuran sampai kondisi *high level,* kemudian lampu indikator *FULL* menyala, pompa 1 dan pompa 2 berenti mengisi bahan baku kedalam tangki.
3. Setelah semua bahan baku berada dalam tangki pencampuran, maka bahan baku diaduk dan dipanaskan hingga suhu yang ditentukan. Pada proses ini lampu indikator *STANDBY* menyala.
4. Setelah bahan baku diaduk dan dipanaskan hingga suhu yang ditentukan, maka lampu indikator *FULL* dan *STANDBY* akan mati, serta pompa 3 aktif menguras tangki sampai kondisi *low level*.

2.1.2 Teh Botol Sosro

Teh botol sosro merupakan salah satu produk dari PT. Sosro. Produk Teh Botol Sosro terbuat dari seduhan teh wangi melatidan dicampur dengan sirup gula cair yang diperoleh dengan melarutkan gula pasir putih. Berikut tahap-tahap pembuatan Teh Botol Sosro pada PT. SOSRO.

1. Penyeduhan Teh

   Teh wangi melati diseduh di dalam tangki ekstraksi dengan air mendidih yang sudah melalui *filter* (penyaring) dan pemanasan. Setelah proses penyeduhan teh selesai, maka teh cair pahit hasil seduhan tersebut dilewatkan ke *filter* cosmos dan dipindahkan ke tangki pencampuran *mixing tank* (tangki pencampuran) hingga mencapai standar yang telah ditentukan menggunakan pompa.

2. Pelarutan Gula

   Gula pasir putih dilarutkan dengan air panas di tangki pelarutan gula sampai menjadi sirup gula. Sirup gula ini kemudian disaring dan dipindahkan kedalam tangki penampungan hingga mencapai standar yang telah ditentukan menggunakan pompa.

3. Pencampuran

   Teh cair pahit dan sirup gula diaduk mencapai standar yang telah ditentukan hingga menjadi teh cair manis.

4. Pemananasan

   Pada proses ini teh cair manis dipanaskan dengan *heat exchanger* (pemanas tidak langsung) hingga mencapai suhu diatas 90°C. Proses pemanasan ini disebut proses pasteurisasi. Pasteurisasi adalah sebuah proses pemanasan makanan dengan tujuan membunuh organisme merugikan seperti bakteri dan virus.

2.2 Motor DC

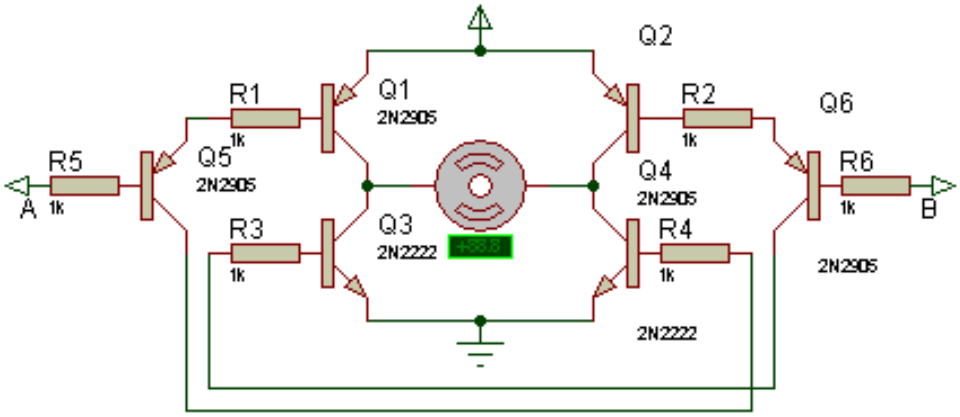
Motor DC dapat berputar karena adanya induksi ketika dihubungkan ke sumber tegangan DC. Arah putaran motor DC tergantung pada polaritas sumber tegangan DC, arah putaran motor DC dapat diatur dengan cara membalikkan input dari polaritas sumber tegangan. Kecepatan putaran motor DC tergantung pada besarnya sumber tegangan, selain itu motor DC dapat juga diatur kecepatannya menggunakan PWM (*Pulse Width Modulation*) atau modulasi lebar pulsa.



Gambar 2.2 Bentuk Fisik Motor DC

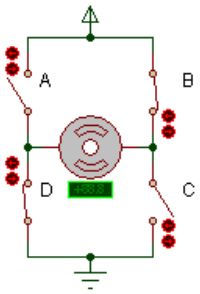
2.3 H-*bridge* (Jembatan-H)

H-Bridge adalah sebuah rangkaian dimana beban (biasanya motor DC atau motor langkah) menjadi titik tengahnya dengan dua jalur yang bisa dibuka tutup untuk melewatkan arus pada motor tersebut, sama seperti huruf "H" (dengan motor DC berada pada garis horizontal) sehingga disebut jembatan-H. Jembatan-H digunakan untuk pengontrol arah putaran motor DC. Pada dasarnya jembatan-H menggunakan transistor sebagai saklar untuk mengontrol arah putaran motor DC.



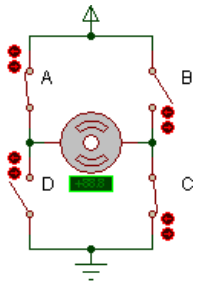
Gambar 2.3 Rangkaian Jembatan-H

Rangkaian jembatan-H dapat dibangun menggunakan transistor yang berfungsi sebagai saklar. Jika A diberi sumber tegangan mengakibatkan motor DC berputar berlawanan arah jarum jam. Ilustrasi transistor sebagai saklar mengakibatkan motor DC berputar searah jarum jam dapat dilihat pada gambar dibawah ini.



Gambar 2.4 Jembatan-H untuk Mengontrol Arah Putaran Berlawanan Arah Jarum Jam

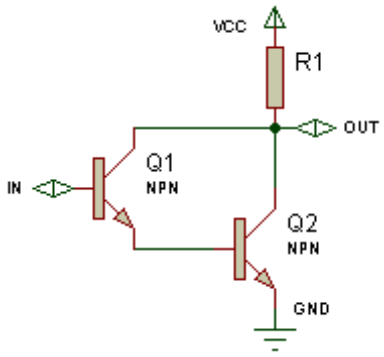
Seperti terlihat pada diatas, jika B diberi sumber tegangan mengakibatkan motor DC berputar searah jarum jam. Ilustrasi transistor sebagai saklar mengakibatkan motor DC berputar searah jarum jam dapat dilihat pada gambar dibawah ini.



Gambar 2.5 Jembatan-H untuk Mengontrol Arah Putaran Motor Searah Jarum Jam

2.4 Transistor Darlington

Transistor Darlington adalah rangkaian elektronika yang terdiri dari sepasang transistor bipolar yang tersambung secara seri. Sambungan seri seperti ini dipakai untuk mendapatkan penguatan (*gain*) yang tinggi, karena hasil penguatan pada transistor yang pertama akan dikuatkan lebih lanjut oleh transistor kedua.



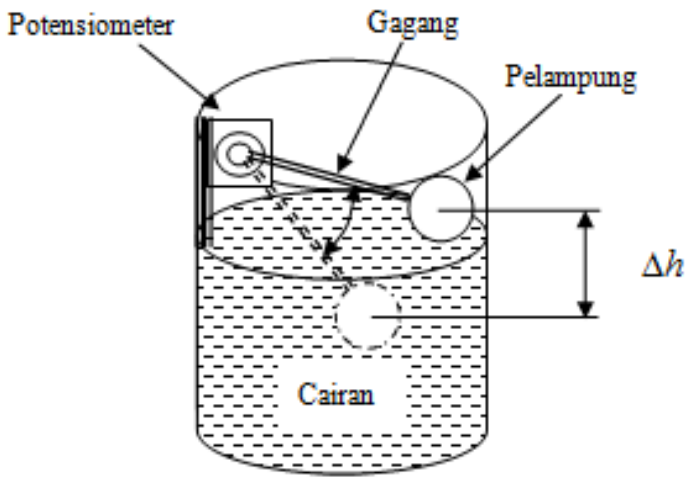
Gambar 2.6 Rangkaian Transistor Darlington

2.5 Sensor Tingkat Ketinggian

Pengukuran tingkat ketinggian dapat dilakukan dengan bermacam cara antara lain dengan pelampung atau *displace*r, gelombang udara, resistansi, kapasitif, ultrasonic, optic, Tehrmal, tekanan, sensor permukaan dan radiasi. Pemilihan sensor yang tepat tergantung pada situasi dan kondisi sistem yang akan digunakan.

2.5.1 Sistem Pelampung

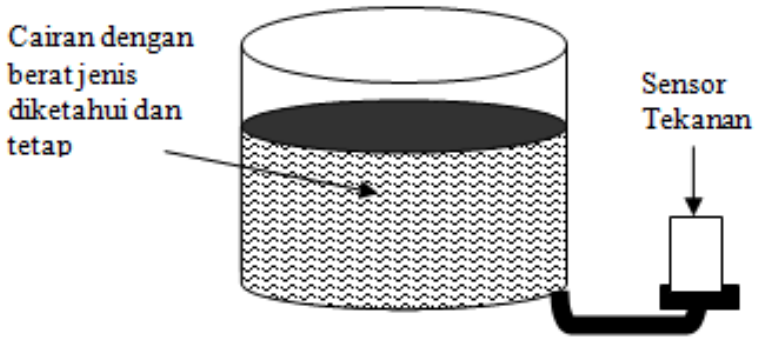
Cara yang paling sederhana dalam mengetaui tingkat ketinggian cairan adalah dengan menggunakan pelampung yang diberi gagang. Pembacaan dapat dilakukan dengan memasang sensor posisi misalnya *potensiometer* pada bagian engsel gagang pelampung. Cara ini cukup baik diterapkan untuk tangki air yang tidak terlalu tinggi.



Gambar 2.7 Sensor Tingkat Ketinggian Menggunakan Pelampung

2.5.2 Sistem Tekanan

Untuk mengukur tingkat ketinggian cairan dapat pula dilakukan menggunakan sensor tekanan yang dipasang di bagian dasar dari tabung. Cara ini cukup praktis, akan tetapi ketelitiannya sangat tergantung dari berat jenis dan suhu cairan sehingga kemungkinan kesalahan pembacaan cukup besar.



Gambar 2.8 Sensor Tingkat Ketinggian Menggunakan Sensor Tekanan

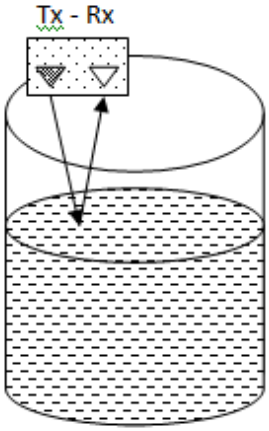
2.5.3 Sensor Ultrasonik

Untuk mengukur tingkat ketinggian cairan dapat pula dilakukan menggunakan sensor ultrasonik yang dipasang di bagian atas dari tabung. Sensor ini terdiri dari pemancar dan penerima.

Pemancar ultrasonik memancarkan gelombang ultrasonik, ketika gelombang ultrasonik mengenai cairan, maka sinyal ini dipantulkan, dan diterima oleh penerima ultrasonik. Selama menunggu pantulan, pengirim ultrasonik akan menghasilkan sebuah pulsa. Pulsa ini akan berhenti ketika gelombang pantulan terdeteksi oleh penerima ultrasonik. Oleh karena itulah lebar pulsa tersebut dapat merepresentasikan jarak antara sensor dengan objek.



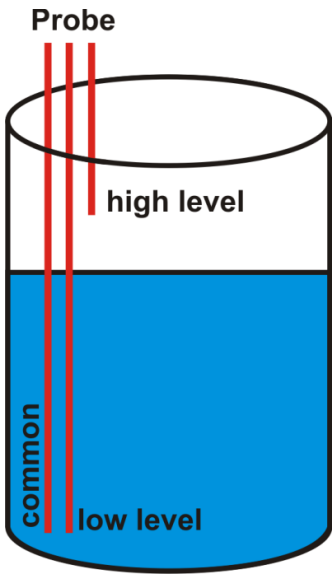
Gambar 2.9 Bentuk Fisik Ping))) Paralac



Gambar 2.10 Sensor Tingkat Ketinggian Menggunakan Sensor Ultrasonik

2.5.4 Sensor *Probe*

Sensor *probe* menggunakan kawat tembaga yang sebagai *probe*-nya. *Common probe* terhubung dengan *ground* sehingga berlogika 0, sedangkan *probe low level* dan *high level* terhubung dengan Vcc sehingga berlogika 1.



Gambar 2.11 Konfigurasi Sensor *Probe*

Apabila *common probe* dan *low level* ataupun *high level* menyentuh air, maka terjadi perubaan logika *probe low level* yang semula berlogika 1 menjadi 0. Perubahan logika ini yang dimanfaatkan sebagai pendeteksian ketinggian air.

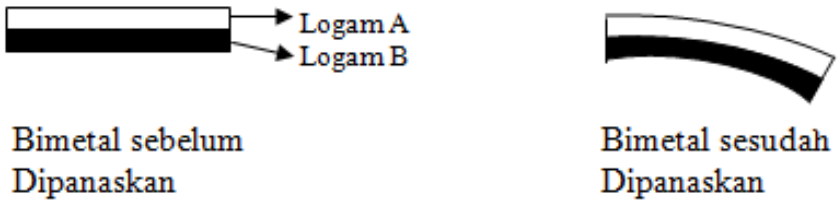
2.6 Sensor Suhu

Sensor suhu adalah alat yang digunakan untuk mengubah suhu menjadi besaran listrik yang dapat dengan mudah dianalisis besarnya. Berikut penjelasan mengenai sensor suhu.

2.6.1 Bimetal

Bimetal adalah sensor suhu yang terbuat dari dua buah lempengan logam yang berbeda koefisien muainya (α) yang direkatkan menjadi satu. Bila suatu logam dipanaskan maka akan terjadi pemuaian, besarnya pemuaian tergantung dari jenis logam dan tingginya temperatur kerja logam tersebut.

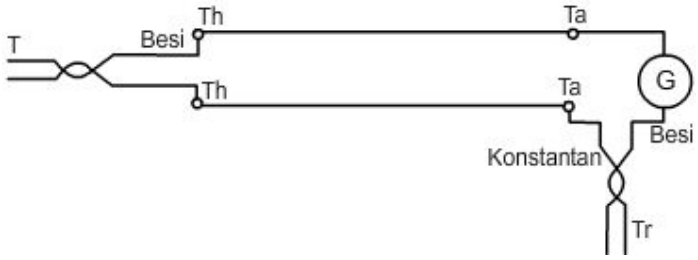
Bila dua lempeng logam saling direkatkan dan dipanaskan, maka logam yang memiliki koefisien muai lebih tinggi akan memuai lebih panjang sedangkan yang memiliki koefisien muai lebih rendah memuai lebih pendek. Oleh karena perbedaan reaksi muai tersebut maka bimetal akan melengkung kearah logam yang muainya lebih rendah. Dalam aplikasinya bimetal dapat dibentuk menjadi saklar *Normally Closed* (NC) atau *Normally Open* (NO).



Gambar 2.12 Kontruksi Bimetal

2.6.2 Termokopel

Termokopel dapat mengukur suhu 3000F sampai dengan suhu tinggi yang digunakan pada proses industri baja, gelas dan keramik yang lebih dari 30000F. Termokopel dibentuk dari dua buah penghantar yang berbeda jenisnya (besi dan konstantan) dan dililit bersama.



Gambar 2.13 Rangkaian Dasar Termokopel

Jika salah satu bagian pangkal lilitan dipanasi, maka pada kedua ujung penghantar yang lain akan muncul beda potensial (emf). Termokopel ditemukan oleh Thomas Johan Seebeck tahun 1820 dan dikenal dengan Efek *seebeck*. Efek *seebeck* adalah sebuah rangkaian termokopel sederhana dibentuk oleh 2 buah penghantar yang berbeda jenis (besi dan konstan) dililit bersama-sama. Salah satu ujung T merupakan *measuring junction* dan ujung yang lain sebagai *reference junction*. *Reference junction* dijaga pada suhu konstan 320F (00C atau 680F (200C). Bila ujung T dipanasi hingga terjadi perbedaan suhu terhadap ujung Tr, maka pada kedua ujung penghantar besi dan konstantan pada pangkal Tr terbangkit beda potensial (*electro motive force*/emf) sehingga mengalir arus listrik pada rangkaian tersebut.

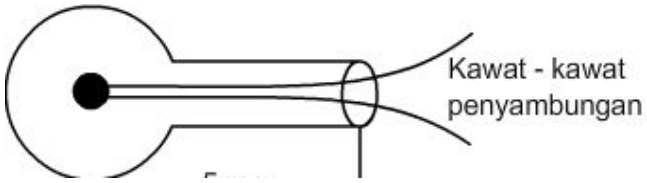
Kombinasi jenis logam penghantar yang digunakan menentukan karakteristik linier suhu terhadap tegangan. Tipe-tipe kombinasi logam penghantar termokopel, yaitu tipe E (*kromel-konstantan*), tipe J (besi-konstantan), tipe K (kromel-alumel), tipe R-S (platinum-platinum rhodium) dan tipe T (tembaga-konstantan).



Gambar 2.14 Bentuk Fisik Termokopel

2.6.3 Thermistor (*Thermal Resistor*)

Thermistor mengubah suhu menjadi resistansi/hambatan listrik yang berbanding terbalik dengan perubahan suhu. Semakin tinggi suhu, semakin kecil resistansi.



Gambar 2.15 Konstruksi Thermistor

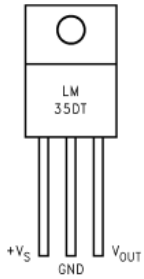
Thermistor dibentuk dari bahan oksida logam campuran, kromium, kobalt, tembaga, besi atau nikel. Bentuk Thermistor :

1. butiran digunakan pada suhu > 7000C dan memiliki nilai resistansi 100 Ω hingga 1 MΩ,
2. keping digunakan dengan cara direkatkan langsung pada benda yang diukur panasnya,
3. batang digunakan untuk memantau perubahan panas pada peralatan elektronik, mempunyai resistansi tinggi dan disipasi dayanya sedang. Thermistor dibuat sekecil-kecilnya agar mencapai kecepatan tanggapan (respon time) yang baik.

Ada dua macam thermistor secara umum: posistor atau PTC (*Positive Temperature Coefficient*), dan NTC (*Negative Temperature Coefficient*). Nilai tahanan pada PTC akan naik jika perubahan suhunya naik, sementara sifat NTC kebalikan dari PTC.

2.6.4 LM35

Sensor LM35 ini merupakan sensor yang banyak digunakan dalam melakukan pengukuran dan pengontrolan suhu, dikarenakan sensor LM35 ini memiliki keakuratan yang tinggi, kemudahan perancangan jika dibandingkan dengan sensor suhu yang lain serta memiliki karakteristik sensor yang cukup baik.



Gambar 2.16 Konfigurasi *Pin* LM35

Berikut karakterisitik LM35.

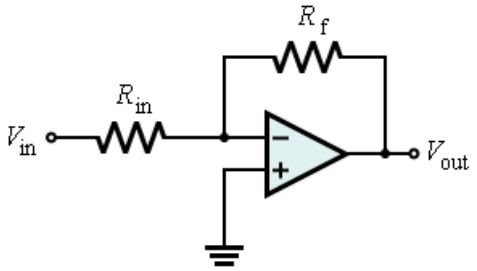
1. Dapat dikalibrasi langsung ke dalam besaran celcius karena sensor suhu jenis LM35 ini sudah difungsikan untuk mendeteksi besaran suhu dalam skala celcius.
2. Memiliki sensitivitas suhu dengan faktor skala linier antara tegangan dan suhu 10 mVolt/ºC, sehingga sensor akan melakukan penginderaan pada saat perubahan suhu setiap suhu 1ºC akan menunjukan tegangan sebesar 10 mV. Karena itu LM35 tidak lagi memerlukan kalibrasi eksternal.
3. Memiliki tingkat kelinieran yang tinggi, yaitu suhu akan naik 1ºC setiap kenaikan 10 mV dan suhu akan turun setiap pengurangan 10mV.
4. Mudah dalam penempatan, yaitu pada penempatannya LM35 dapat ditempelkan dengan perekat atau dapat pula ditempatkan pada permukaan semen akan tetapi suhunya akan sedikit berkurang sekitar 0,01 ºC karena terserap pada suhu permukaan tersebut.
5. Memiliki pemanasan sendiri yang rendah (low self heating) yaitu kurang dari 0,5ºC,sehingga tingkat kesalahan dalam pembacaan suhu rendah.
6. Memiliki jangkauan maksimal operasi suhu antara -55 ºC sampai +150 ºC.
7. Bekerja pada tegangan 4 sampai 30 volt dengan arus rendah yaitu 60 µA.

2.7 Penguatan Operasional (*Op-Amp*)

Op-Amp merupakan kumpulan puluhan transistor dan resistor dalam bentuk satu chip IC. *Op-Amp* merupakan komponen aktif linear yang merupakan penguat gandeng langsung (*direct coupling*), dengan penguatan terbuka (*open gain*) yang sangat besar dan dapat dipakai untuk menjumlahkan, mengalikan, membagi, mendiferensialkan, serta mengintegralkan tegangan listrik. IC *Op-Amp* sering digunakan untuk perhitungan-perhitungan analog, instrumentasi, maupun berbagai macam aplikasi kontrol.

2.7.1 Penguat Operasional Membalik (*Inverting Op-Amp*)

Inverting op-amp akan menguatkan tegangan pada masukan serta membalik hasil penguatan tersebut, jadi keluaran dari rangkaian ini akan selalu memiliki polaritas yang berlawananan dengan sinyal masukannya.



Gambar 2.17 Rangkaian *Inverting Op-Amp*

Penguatan tegangan (A_v) pada rangkaian inverting op-amp diatas, ditentukan oleh persamaan berikut :

$$A_v = -{R_f}/{R_{i\,n}} \quad \text{.....(2.1)}$$

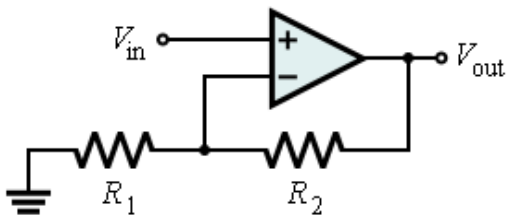
Tegangan keluaran (V_{out}) diperoleh dengan cara mengalikan tegangan masukan (V_{in}) yang diketahui dengan faktor penguatan, atau menggunakan persamaan sebagai berikut :

$$V_{out} = -\frac{R_f}{R_{in}} V_{in} \quad \text{.....(2.2)}$$

Tanda minus hanya menunjukkan bahwa keluaran berlawanan fasa terhadap masukannya.

2.7.2 Penguat Operasional Tak Membalik (*Non-Inverting Op-Amp)*

Dalam konfigurasi *non-inverting op-amp*, umpan balik digunakan untuk mengatur penguatan tetap diberikan pada masukan membalik, tetapi V_{in} diberikan pada masukan tak membalik sehingga tegangan keluaran akan selalu sefasa dengan tegangan masukannya.



Gambar 2.18 Rangkaian *Non-Inverting Op-Amp*

Penguatan tegangan (A_v) pada rangkaian non-inverting op-amp diatas, ditentukan oleh persamaan berikut :

$$A_v = \frac{R_f}{R_{in}} + 1 \quad \text{.....(2.3)}$$

Tegangan keluaran (V_{out}) diperoleh dengan cara mengalikan tegangan masukan (V_{in}) yang diketahui dengan faktor penguatan, atau menggunakan persamaan sebagai berikut :

$$V_{out} = \left({R_f}/{R_{in}} + 1 \right) V_{in} \quad \text{.....(2.4)}$$

2.8 Saluran Serial Komputer

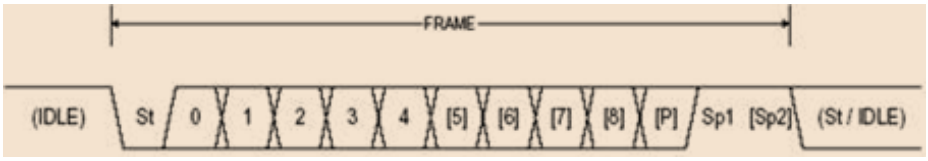
Ada 2 macam cara komunikasi data serial, yaitu :

8. komunikasi serial sinkron atau *Universal Synchronous serial Receiver and Transmitter* (USART) dan
9. komunikasi serial sinkron atau *Universal Asynchronous serial Receiver and Transmitter* (UART).

Pada komunikasi data serial sinkron, *clock* dikirimkan bersama sama dengan data serial, tetapi *clock* tersebut dibangkitkan oleh masing-masing sisi pengirim maupun penerima. Sedangkan pada komunikasi serial asinkron tidak diperlukan *clock* karena data dikirimkan dengan kecepatan tertentu yang sama baik pada pengirim dan penerima.

Saluran serial IBM PC termasuk jenis komunikasi serial asinkron. Agar terjadi sinkronisasi antara pengirim dan penerima, maka al-hal yang harus dilakukan adalah sebagai berikut.

10. Menentukan standar *baud rate* (kecepatan transfer bit data *frame* dalam 1 detik). *Baud rate* dapat dipilih bebas dalam rentang tertentu. *Baud rate* yang umum dipakai adalah 110, 135, 150, 300, 600, 1200, 2400, dan 9600 (bit/detik). Dalam komunikasi data serial, *baud rate* dari kedua alat yang berhubungan harus diatur pada kecepatan yang sama.
11. Setiap *byte* data akan diberikan bit-bit tambahan yang disebut dengan *frame* (bingkai), yang terdiri dari *start bit* yang menandakan dimulainya sebuah transmisi, dan *stop bit* yang menandakan akhir dari transmisi.
12. *Start bit* selalu berlogika rendah.
13. *Stop bit* selalu berlogika tinggi.
14. Dapat menambahkan dengan *bit parity*. *Bit parity* (paritas) adalah *bit* yang digunakan untuk menandakan data memiliki jumlah *bit* berlogika 1 sebanyak ganjil atau genap. Paritas dibutuhkan untuk mengetahui jika ada transfer *byte* yang cacat, dan menghasilkan data yang tidak sesuai dengan *bit* paritasnya.



Gambar 2.19 Sinyal Komunikasi IBM-PC

2.8.1 Karakteristik Sinyal Saluran Serial

Standar sinyal komunikasi serial yang banyak digunakan adalah standar RS232 yang dikembangkan oleh *Electronic Industri Association* (EIA). Standar ini hanya menyangkut komunikasi antara *Data Terminal Equipment* (DTE) dengan alat-alat pelengkap komputer *Data Circuit Equipment* (DCE). Standar sinyal RS232 memiliki ketentuan tingkat ketinggian tegangan sebagai berikut :

15. Logika '1' disebut '*mark*' terletak antara -3 Volt sampai -25 Volt,
16. Logika '0' disebut '*space*' terletak antara +3 Volt sampai +25 Volt, dan
17. Daerah tegangan antara -3 Volt sampai +3 Volt adalah *invalid level*, yaitu daerah tegangan yang tidak memiliki logika pasti sehingga harus dihindari. Demikian juga level tegangan dibawah -25 Volt dan diatas +25 Volt juga harus dihindari karena bisa merusak *line driver* pada saluran RS232.

2.8.2 Konfigurasi Saluran Serial

Konektor DB9 pada bagian belakang komputer adalah *port* serial RS232 yang biasa dinamai dengan COM1 dan COM2.



Gambar 2.20 Konektor DB9

Tabel 2.1 Keterangan Sinyal dan Kaki Saluran Serial DB9

Nomor Kaki Konektor DB9	Nama Sinyal	Arah Sinyal
1	DCD (*Data Carrier Detect*)	Dari DCE
2	RD (*Received Data*)	Dari DCE
3	TD (*Transmitted Data*)	Dari DTE
4	DTR (DTE *Ready*)	Dari DTE
5	*Ground*	-
6	DSR (DCE *Ready*)	Dari DCE
7	RTS (*Request to Send*)	Dari DTE
8	CTS (*Clear to Send*)	Dari DCE
9	RI (*Ring Indikator*)	Dari DCE

Keterangan mengenai fungsi saluran pada konektor DB9 adalah sebagai berikut:

18. *Data Carrier Detect*, dengan saluran ini DCE memberitahukan ke DTE bahwa pada konektor masukkan ada data masuk.
19. *Receive Data*, digunakan DTE menerima data dari DCE.
20. *Transmitted Data*, digunakan DTE mengirimkan data ke DCE.
21. *Data Terminal Ready*, pada saluran ini DTE memberitahukan kesiapan konektornya.
22. *Signal Ground* merupakan saluran *ground*.
23. DCE *Ready*, sinyal aktif pada saluran ini menunjukkan bahwa DCE sudah siap.
24. *Request To Send*, dengan saluran ini DCE diminta mengirim data oleh DTE.
25. *Clear To Send*, dengan saluran ini DCE memberitahukan bahwa DTE boleh mulai mengirim data.

26. *Ring Indikator*, pada saluran ini DCE memberitahukan ke DTE bahwa sebuah stasiun menghendaki berhubungan dengannya.

Untuk dapat menggunakan saluran serial harus diketahui dahulu alamat dari saluran serial tersebut. Biasanya tersedia dua saluran serial pada CPU, yaitu COM1 dan COM2. COM1 memiliki alamat 1016 (3F8h) dan COM2 memeiliki alamat 760 (2F8h).

Tabel 2.2 *Register* dan Alamat *Register* Saluran Serial

Nama *Register*	COM1	COM2
TX *Buffer*	3F8h	2F8h
RX *Buffer*	3F8h	2F8h
Baud Rate Divisor Latch (LSB)	3F8h	2F8h
Baud Rate Divisor Latch (MSB)	3F9h	2F9h
Interrupt Enable Register	3F9h	2F9h
Interrupt Identification Register	3FAh	2FAh
Line Control Register	3FBh	2FBh
Modem Control Register	3FCh	2FCh
Line Status Register	3FDh	2FDh
Modem Status Register	3FEh	2FEh

2.8.3 *Flow Control*

Jika *baud rate* dari DTE ke DCE (misal dari komputer ke modem) lebih cepat dari pada pemindahan data dari DCE ke DCE (modem ke modem), maka cepat atau lambat kehilangan data akan terjadi karena *buffer* (penyangga) pada DCE akan mengalami *overflow*. Untuk itu diperlukan sistem *flow control* untuk mengatasi masalah tersebut. Ada 2 macam *flow control* yaitu secara perangkat keras dan secara perangkat lunak.

27. *Flow control* secara perangkat keras atau sering disebut RTS/CTS, menggunakan dua kabel untuk melakukan pengontrolan. Komputer akan mengatur (*set*) saluran *Request to Send* (RTS), jika akan mengirimkan data ke DCE. Jika penyangga di DCE siap menerima data, maka DCE akan membalas dengan mengatur saluran *Clear to Send* (CTS) dan komputer akan mulai mengirimkan data. Jika penyangga telah penuh, maka saluran akan kembali ke pengaturan semula (*reset*) dan komputer akan menghentikan pengiriman data sampai saluran ini diatur kembali.
28. *Flow control* secara perangkat lunak atau yang sering disebut dengan Xon (karakter ASCII 17) dan Xoff (karakter ASCII 19). DCE akan mengirimkan Xoff ke komputer untuk memberitahukan agar komputer menghentikan pengiriman data apabila penyangga pada DCE telah penuh. Jika penyangga telah kembali siap menerima data, DCE akan mengirimkan karakter Xon ke komputer dan komputer akan melanjutkan pengiriman data sampai data terkirim semua.

2.9 LabVIEW

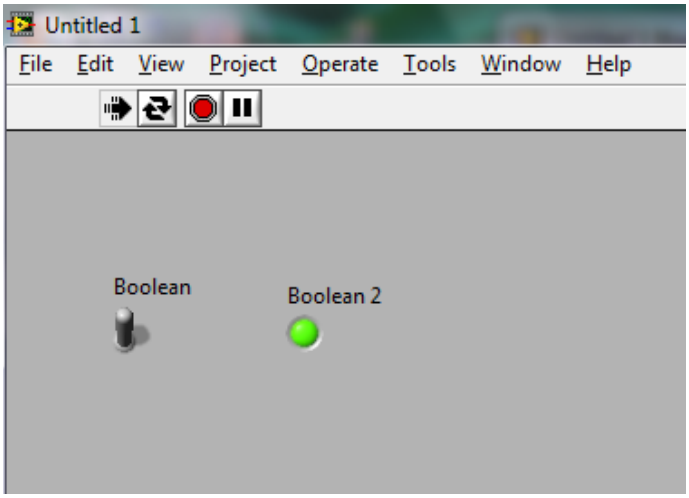
LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) adalah sebuah *software* pemrograman yang diproduksi oleh National Instrument. LabVIEW menggunakan bahasa pemrograman grafis atau blok diagram, berbeda dengan Visual Basic, C++, Borland Delphi, ataupun Matlab yang menggunakan bahasa pemrograman menggunakan teks. LabVIEW berisi seperangkat peralatan untuk memperoleh, menganalisis, menampilkan, dan menyimpan data, dan juga

sebagai alat untuk memecahkan masalah kode yang dapat digunakan untuk membangun berbagai aplikasi di industri, seperti akuisisi data, sistem instrumen dan control, telekomunikasi, manufaktur, otomotif, dan lain-lain.

LabVIEW disebut juga *Virtual Instrument* atau VI, disebut VI karena tampilan program LabVIEW dan operasi meniru instrumen fisik. Dalam pemrogramannya LabVIEW terdapat *Front Panel* dan *Block Diagram*.berikut penjelasan mengenai *Front Panel* dan *Block Diagram*.

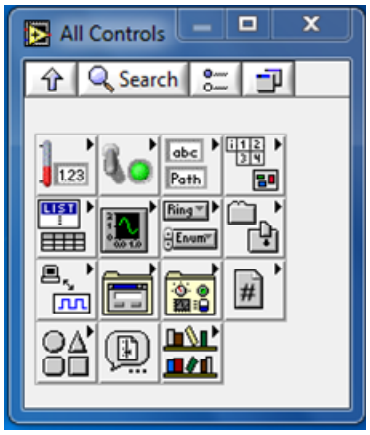
2.9.1 *Front Panel*

Front Panel merupakan salah satu dari dua jendela LabVIEW digunakan untuk membangun sebuah VI selain jendela *Block Diagram*. *Front Panel* berisi *control* dan *indikator*. *Control* berfungsi untuk mensimulasikan perangkat input instrumen, sedangkan *indikator* berfungsi untuk mensimulasikan perangkat output instrumen.



Gambar 2.21 *Front Panel*

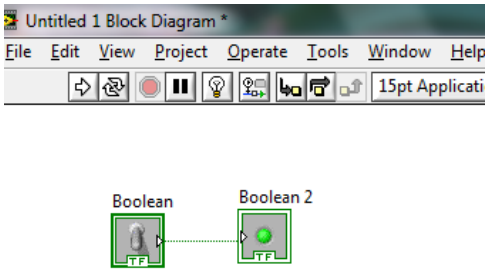
Control Palette hanya tersedia pada *Front Panel*. *Control Palette* berisi *control* dan *indikator* digunakan untuk membangun suatu *Front Panel*. *Control* dan *indikator* tersebut telah tersedia di *subpalettes* yang tersusun berdasarkan jenis *control* dan *indikator*.



Gambar 2.22 *Controls Palette*

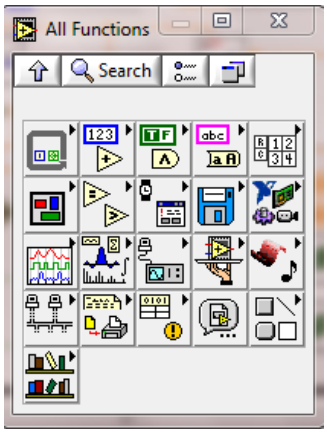
2.9.2 *Block Diagram*

Setelah membangun *Front Panel*, *programmer* dapat menambahkan kode-kode representasi grafis pada *Block Diagram*. Seperti halnya baris teks yang ditemukan dalam bahasa C atau BASIC, kode-kode tersebut berfungsi untuk mengontrol objek pad *Front Panel*.



Gambar 2.23 *Block Diagram*

Function Palette hanya tersedia pada *Block Diagram*. *Function Palette* berisi fungsi yang terletak di subpalettes yang digunakan untuk membangun *Block Diagram*.

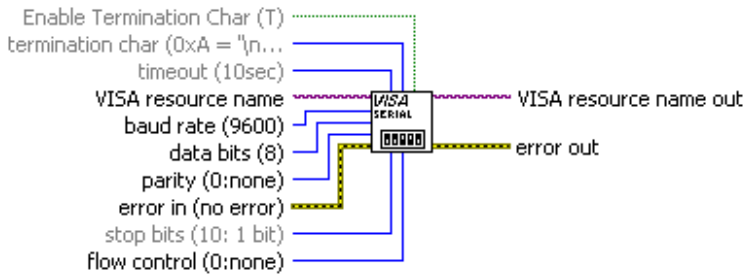


Gambar 2.24 *Function Palette*

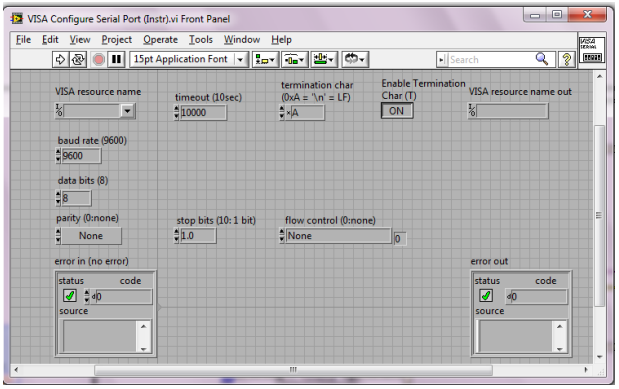
2.9.3 Komunikasi Serial LabVIEW

LabVIEW dapat melakukan komunikasi dengan peralatan lain. Melalui *port* serial, LabVIEW dapat melakukan komunikasi kemudian mengontrol ataupun memantau peralatan yang terhubung. Pada *function pallete* telah tersedia *sub-vi* yang memudakan dalam menggunakan *file-file* LabVIEW. *Sub-vi* adalah anak dari LabVIEW atau dianalogikan seperti sub bab dalam bab.

Pada *function pallete* tersedia VISA *serial configure serial port* (Instr).vi. VISA *serial configure serial port* (Instr).vi berfungsi untuk menentukan inisialisasi *baudrate*, *parity*, *data bit*, *stop bit*, dan sebagainya. VISA *serial configure serial port* (Instr).vi dapat diperoleh pada *Function pallete*➔ *Instrument* I/O ➔ Serial ➔ VISA *serial configure serial port*.



Gambar 2.25 Tampilan *Icon Visa Serial Configure Serial Port.Vi* pada *block diagram* LabVIEW



Gambar 2.26 Tampilan *Visa Serial Configure Serial Port.Vi* pada *Front Panel*

Keterangan :

29. *Enable* Termination *Char*

Berfungsi untuk mempersiapkan perangkat serial untuk *termination char.* Jika *enable termination* berlogika *true (default),* maka VI_ATTR_ASRL_END_IN berarti peralatan serial mengenali *termination char*. Jika *false* berarti VI_ATTR_ASRL_END_IN berlogika 0 (*None*) dan peralatan serial tidak mengenali *termination char.*

30. *Termination Char*

Merupakan pemanggilan untuk pembacaan operasi. Operasi pembacaan karakter berakhir ketika *terminasi char* dibaca dari perangkat serial. 0xA setara dengan karakter *linefeed* (\n). 0xD adalah pesan *string* pada saat termination char tidak dibaca lagi, yang setara dengan *carriage return*(/r).

31. *Timeout*

Berfungsi untuk menntukan waktu dalam milidetik untuk operasi penulisan maupun pembacaan. *Timeout* memiliki nilai default 10000.

32. VISA *resource name*

Berfungsi untuk menentukan alamat *port*.

33. *Baud rate*

Baud rate adalah tingkat kecepatan transfer data. *Baud rate* memiliki nilai *default* 9600.

34. *Data bits*

Data bits adalah jumlah bit saat data masuk atau terbaca. Nilai data bits antara 5 sampai 8. Nilai *default* data bits adalah 8.

35. *Parity*

Parity berfungsi untuk menentukan paritas digunakan untuk setiap *frame* yang akan dikirim atau diterima.

2.10 ATmega8535

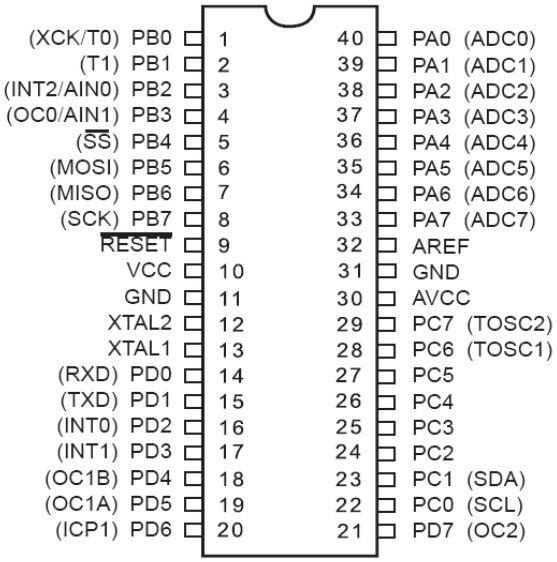
2.10.1 Pengenalan ATmega8535

ATmega8535 merupakan salah satu mikrokontroler 8 *bit* buatan Atmel untuk keluarga AVR yang diproduksi secara masal pada tahun 2006. Karena merupakan keluarga AVR, maka ATmega8535 juga menggunakan arsitektur RISC (*Reduced Instruction Set Computing).*

AVR memilki keunggulan dibandingkan dengan mikrokontroler lain, keunggulan mikrokontroler AVR yaitu memiliki kecepatan dalam mengeksekusi program yang lebih cepat, karena sebagian besar instruksi dieksekusi dalam 1 siklus *clock* (lebih cepat dibandingkan mikrokontroler keluarga MCS51 yang memiliki arsitektur *Complex Intrukstion Set Computer* (CISC)). ATmega8535 memiliki fitur-fitur , antara lain :

36. sistem mikroprosesor berbasis RISC dengan kecepatan maksimal 16 Mhz,
37. memiliki *memory flash* untuk program sebesar 8 Kb,
38. SRAM sebesar 512 *byte*,
39. EEPROM sebesar 512 *byte*,
40. 2 buah *Timer/Counter* 8 *bit* dan 1 buah *Timer/Counter* 16 *bit*,
41. 4 *channel* PWM,
42. 8 *channel* ADC dengan resolusi 10 *bit*,
43. komunikasi USART.

2.10.2 Konfigurasi *Pin*



Gambar 2.27 Konfigurasi *Pin* ATmega8535

Gambar diatas merupakan konfigurasi pin ATmega8535 dengan kemasan 40 *pin* DIP (*Dual In-line Package*). Berikut penjelasan fungsi-fungsi pin ATmega8535 :

44. VCC merupakan *pin* yang berfungsi sebagai *pin* masukan catu daya.
45. GND merupakan pin *ground*.
46. PORTA (PA0..PA7) merupakan *pin* I/O dua arah dan *pin* masukan ADC
47. PORTB (PB0..PB7) merupakan *pin* I/O dua arah dan *pin* fungsi khusus, yaitu *timer/counter*, *comparator analog*,dan SPI.
48. PORTC (PC0..PC7) merupakan *pin* I/O dua arah dan *pin* fungsi khusus, yaitu TWI, *comparator analog* dan *timer oscillator*.
49. PORTD (PD0..PD7) merupakan *pin* I/O dua arah dan *pin* fungsi khusus, yaitu *comparator analog*, interupsi eksternal, dan komunikasi serial.

50. RESET merupakan *pin* yang digunakan untuk me-*reset* mikrokontroler.
51. XTAL1 dan XTAL2 merupakan *pin* masukan *clock* ekstenal.
52. AVCC merupakan *pin* masukan tegangan untuk ADC.
53. AREF merupakan *pin* masukan tegangan referensi ADC.

2.10.3 Dasar I/O

ATmega8535 memiliki 4 buah PORT I/O dinamakan PORTA, PORTB, PORTC, dan PORTD. Masing-masing PORT memiliki 3 *register* PORTx, DDRx dan PINx (x = urutan, misalnya PORTA, DDRA, PINA). Masing-masing *register* dapat diaturdan dibaca tiap *bit*-nya. DDRx merupakan *register* pengaturan arah PORT, sedangkan PORTx digunakan untuk mengeluarkan dan PINx untuk membaca dari PORT.

Tabel 2.3 Pengaturan PORT

	DDR *bit* = 1	DDR *b*it = 0
PORT *bit* = 1	*Output High*	*Input Pull-up*
PORT *bit* = 0	*Output Low*	*Input floating (Tri-state)*

2.10.4 ADC

ATMega8535 telah dilengkapi dengan 8 *channel* 10-*bit Analog to Digital Converter* (ADC). Rangkaian internal ADC ini memiliki catu daya tersendiri yaitu kaki AVCC. Data pembacaan ADC terdapat pada *register* ADCH dan ADCL.

Sedangkan untuk pengaturan digunakan *register* ADMUX, ADCSRA, dan SFIOR.

Tabel 2.4 ADMUX (ADC *Multiplexer Selection Register*)

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

Keterangan :

54. REFS1:0 (*Reference Selection Bits*), untuk mengatur sumber tegangan referensi untuk ADC.
55. ADLAR (*ADC Left Adjust Result*), untuk format penyimpanan data ADC.
56. MUX4:0 (*Analog Channel and Gain Selection Bits*), untuk memilih saluran pembacaan ADC.

Tabel 2.5 ADCSRA (ADC *Control and Status Register* A)

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

Keterangan :

57. ADEN (ADC *Enable*).
58. ADCS (ADC *Start Conversion*). Di-*set* untuk mulai konversi ADC. *Bit* ini akan aktif selama ADC sibuk dan akan bernilai 0 jika telah selesai dikonversi.

59. ADATE (ADC *Auto Trigger Enable*). Mengatur aktivasi picu otomatis operasi ADC. Pengaturan sumber pemicuan terdapat pada *register* SFIOR.
60. ADIF (ADC *Interrupt Flag*)**.** Bernilai 1 jika konversi ADC telah selesai. Pengaktifan bit ini juga akan memicu interupsi jika ADIE di-*set*.
61. ADIE (ADC *Interrupt Enable*) .
62. ADPS2:0 (ADC *Prescaler Select*). Mengatur *prescaler* untuk ADC.

ATmega8535 mengubah tegangan atau data analog ke digital dengan rumus sebagai berikut :

$$ADC = {V_{in} \times n}/{V_{ref}} \quad \text{.....(2.5)}$$

Keterangan :

ADC : hasil konversi data analog ke digital

Vin : tegangan input analog

n : jumlah resolusi (8 *bit* atau 10 *bit*)

Vref : tegangan referensi

Berikut contoh pemrograman ADC dan hasil pembacaan ADC ditampilkan pada led :

```
#include <mega8535.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x60
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for Teh stabilization of Teh ADC input voltage
delay_us(10);
// Start Teh AD conversion
ADCSRA|=0x40;
```

```
// Wait for Teh AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
void main(void)
{
PORTC=0xff;
DDRC=0xff;
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
SFIOR&=0xEF;
while (1)
    {
    // Place your code here
    read_adc(0);
    PORTC=ADCH;   };}
```

2.10.5 *Timer/Counter*

ATmega8535 Memiliki 3 buah *timer/counter*, yaitu *timer/counter0* (8-*bit*), *Timer/counter1* (16-*bit*) dan *timer/counter2* (8-*bit*). Penjelasan untuk masing-masing *timer/counter* akan dijelaskan berikut ini :

2.10.5.1 *Timer/Counter0* dan *Timer/Counter2*

Timer/counter0 dan *timer/counter2* adalah 8-*bit timer/counter* yang serbaguna. Kedua *timer/counter* tersebut memiliki fitur antara lain :

63. 1 *channel counter*,
64. *timer* dinolkan (*clear*) pada *match-compare* (*auto reload*),
65. *glitch-free*, *phase correct pulse width modulator* (PWM),
66. *counter* yang di-*trigger* dari luar (hanya pada *timer/counter0*),

67. 10-*bit clock prescaler*,

68. Interupsi pada saat *overflow* (TOVn) dan *match compare* (OCFn).

Register TCNTn (*Timer/Counter Register*) merupakan *register* yang menyimpan nilai dari *timer/counter*. *Register* OCRn (*Output Compare Register*) merupakan *register* pembanding, jika nilai OCRn = TCNTn maka terjadi *Compare Match*. *Compare Match* dapat digunakan untuk menghasilkan pulsa pada *pin* OCn. Pengaturan *timer/counter0* dan *timer/counter2* dilakukan melalui *register Timer/Counter Control Register* (TCCRn).

Tabel 2.6 TCCRn (*Timer/Counter Control Register* n)

Bit	7	6	5	4	3	2	1	0	
	FOCn	WGMn0	COMn1	COMn0	WGMn1	CSn2	CSn1	CSn0	TCCRn
Read/Write	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	

Keterangan :

69. FOCn (*Force Output Compare*)

Memaksakan operasi *Compare Match*. Hanya aktif pada *mode* non-PWM. FOCn tidak akan memicu terjadinya interupsi atau men-*clear*-kan *timer* pada *mode* CTC.

70. WGMn1:0 (*Waveform Generation Mode*)

Bit-bit ini mengontrol kenaikkan dari *counter*, sumber dari nilai maksimal dari *counter* dan tipe *timer/counter* yang akan digunakan.

71. COMn1: 0 (*Compare Match Output*)

    Bit-bit ini mengontrol *pin Output Compare* n (OCn). Apabila kedua *bit* tersebut bernilai 0 maka OCn akan berfungsi sebagai *pin* biasa. Apabila salah satu *bit* bernilai 1 maka fungsi dari OC0 bergantung pada pengaturan pada WGMn1:0.

72. CSn2:0 (*Clock Select*)

    Bit-bit ini mengontrol sumber *clock* yang akan digunakan dan *prescaler*-nya. Terdapat perbedaan konfigurasi t CSn2:0 untuk kedua *timer/counter*.

Sumber *clock* untuk *timer/counter*2 *default*-nya terhubung dengan MCU *clock* (clk). *Timer/counter*2 dapat menggunakan *clock* eksternal yang terhubung dengan *pin* TOSC1 dan TOSC2 dan mengaktifkan *bit* AS2 pada *register Asynchronous Status Register* (ASSR).

2.10.5.2 *Timer/Counter1*

Timer/counter1 merupakan *timer/counter* 16-*bit* yang memungkinkan pewaktuan yang lebih akurat. Fitur dari *Timer/Counter1* antara lain :

73. 16-*bit* (memungkinkan untuk 16-*bit* PWM), 2 buah *compare unit*,
74. 2 buah *register* pembanding,
75. 1 buah *input capture unit* dengan *noise canceller*,
76. *clear timer* pada *match compare* (*auto reload*),
77. *glitch-free, Phase Correct Pulse Width Modulator* (PWM),

78. perioda PWM yang dapat diubah-ubah,

79. generator frekuensi,

80. *counter* yang di-*trigger* dari luar,

81. 4 buah pemicu interupsi, (TOV1, OCF1A, OCF1B, and ICF1).

Register TCNT1H:TCNT1L (*Timer/Counter1 Register*) merupakan *register* 16-*bit* yang menyimpan nilai dari *timer/counter1*. *Timer/counter1* memiliki 2 *register* pembanding OCR1AH:OCR1AL dan OCR1BH:OCR1BL (*Output Compare Register*) merupakan *register* pembanding 16-*bit*.

Selain diatas *timer/counter1* juga memiliki *register* ICR1H:ICR1L (*Input Capture Register*) yang mencacah kejadian pada *pin* ICP1. Pengaturan *timer/counter1* dilakukan melalui 2 buah *register Timer/Counter1 Control Register* (TCCR1A dan TCCR1B).

Tabel 2.7 TCCR1A (*Timer/Counter1 Control Register* A)

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	F0C1A	F0C1B	WGM11	WGM10	TCCR
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Keterangan :

82. COM1A1: 0 (*Compare Match Output Mode* untuk *channel* A)

83. COM1B1: 0 (*Compare Match Output* Mode untuk *channel* B)

84. FOC1A (*Force Output Compare* untuk *channel* A)

85. FOC1B (*Force Output Compare* untuk *channel* B)

Tabel 2.8 TCCR1B (*Timer/Counter1 Control Register* B)

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Keterangan :

86. WGM13:0 (*Waveform Generation Mode*)
87. ICNC1 (*Input Noise Canceler*), diberi nilai 1 untuk mengaktifkan *noise canceler* untuk masukkan *pin* ICP1.
88. ICES (*Input Capture Edge Select*), mengontrol jenis pemicuan yang akan mengaktifkan *capture* pada *pin* ICP1 (0 = *falling edge*, 1 = *rising edge*)
89. CS12:0 (Clock Select), *bit-bit* ini mengontrol sumber *clock* yang akan digunakan dan *prescaler*-nya. Terdapat perbedaan konfigurasi t CSn2:0 untuk kedua *timer/counter*.

Berikut contoh penggunaan *timer* pada ATmega8535, program ini menampilkan *timer* menggunakan led :

```
#include <mega8535.h>
unsigned char led=0xfe;
 void main (void)
{
        DDRC=0xff;   // port C sebagai output
        PORTC=led;
        TCNT0=0x00;// setting inisial counter0
        TCCR0=0x05;     // setting skala clock
        TIMSK=0x01;     // aktifkan interrupt timer0
        TIFR=0x00;      // hapus bendera interrupt timer0
        #asm ("sei");
        while(1)
        {PORTC=led;}
 }
```

```
interrupt [TIM0_OVF] void timer0_overflow(void)
{       TCNT0=0x00;  // setting inisial counter0
        led<<=1;       // geser data led ke kiri 1 kali
        led|=1;        // led di-OR-kan dengan data 1
        if (led==0xff) {led=0xfe;}
        PORTC=led; // keluarkan data led ke port C
}
```

2.10.6 *Interrupt* (Interupsi)

Interupsi adalah kondisi di mana pada saat program utama dieksekusi/dikerjakan oleh CPU kemudian tiba-tiba berhenti untuk sementara waktu karena ada rutin lain yang harus ditangani terlebih dahulu oleh CPU, dan setelah selesai mengerjakan rutin tersebut CPU kembali mengerjakan instruksi pada program utama. ATmega8535 mempunyai beberapa 21 buah sumber interupsi. Untuk mengaktifkan *interrupt bit* I (*Global Interrupt Enable*) pada SREG perlu diaktifkan terlebih dahulu kemudian melakukan pengaturan pada *register* yang bersesuaian untuk masing-masing *interrupt*.

Tabel 2.9 Alamat *Interrupt*

No	Nama	Alamat	Sumber
1	Reset	0x0000	*Reset*
2	INT0	0x0001	*External Interrupt Request 0*
3	INT1	0x0002	*External Interrupt Request 1*
4	TIMER2 COMP	0x0003	*Timer/Counter 2 Compare Match*
5	TIMER2 OVF	0x0004	*Timer/Counter 2 Overflow*
6	TIMER1 CAPT	0x0005	*Timer/Counter 1 Capture Event*
7	TIMER1 COMPA	0x0006	*Timer/Counter 1 Compare Match A*
8	TIMER1 COMPB	0x0007	*Timer/Counter 1 Compare Match B*
9	TIMER1 OVF	0x0008	*Timer/Counter 1 Overflow*
10	TIMER0 OVF	0x0009	*Timer/Counter 0 Overflow*
11	SPI STC	0x000A	*Serial Transfer Complete*
12	USART RXC	0x000B	*USART Rx Complete*
13	USART UDRE	0x000C	*USART Data Register Empty*
14	USART TXC	0x000D	*USART Tx Complete*
15	ADC	0x000E	*ADC Conversion Complete*
16	EE_RDY	0x000F	*EEPROM Ready*
17	ANA_COMP	0x0010	*Analog Comparator*
18	TWI	0x0011	*Two-Wire Serial Interface*
19	INT2	0x0012	*External Interrupt Request 2*
20	TIMER2_COMP	0x0013	*Timer/Counter0 Compare Match*
21	SPM_RDY	0x0014	*Store Program Memory Ready*

ATmega8535 memiliki 3 buah sumber interupsi eksternal (INT0, INT1 dan INT2). Untuk INT0 dan INT1 pengaturan keadaan yang menyebabkan terjadinya interupsi terdapat pada *register* MCUCR *(MCU Control Register),* sedangkan untuk INT2 terdapat pada MCUCSR *(MCU Control and Status Register)*.

Tabel 2.10 MCUCR (MCU *Control Register*)

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	

ISC01 dan ISC00 merupakan *bit* untuk mengatur kondisi terjadi interupsi untuk INT0, sedangkan ISC10 dan ISC11 merupakan *bit* untuk mengatur kondisi terjadinya interupsi untuk INT1.

Tabel 2.11 Pengaturan Kondisi Terjadinya Interupsi pada INT0

ISC11	ISC10	Deskripsi
0	0	Logika 0 pada INT1 menyebabkan interupsi
0	1	Perubahan logika INT1 menyebabkan interupsi
1	0	Transisi dari 1 ke 0 (*falling edge*) pada INT1 menyebabkan interupsi
1	1	Transisi dari 0 ke 1 (*rising edge*) pada INT1 menyebabkan interupsi

Tabel 2.12 MCUCSR (MCU *Control and Status Register*)

Bit	7	6	5	4	3	2	1	0	
	-	ISC2	-	-	WRDF	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						

Tabel 2.13 Pengaturan Kondisi Terjadinya Interupsi pada INT2 ISC2

ISC2	Deskripsi
0	Transisi dari 1 ke 0 (*falling edge*) pada INT2 menyebabkan interupsi
1	Transisi dari 0 ke 1 (*rising edge*) pada INT2 menyebabkan interupsi

Untuk mengaktifkan masing-masing interupsi selain *bit* I pada SREG harus aktif juga dilakukan pengaturan pada *register* GICR *(Global Interrupt Control Register)*.

Tabel 2.14 GICR (*Global Interrupt Control Register*)

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0		

Tabel 2.15 *Bit-Bit* untuk Mengaktifkan Interupsi Eksternal *Bit*

Bit	Deskripsi
INT1	*External Interrupt Request* 1 *Enable* (diberi nilai 1 untuk mengaktifkan)
INT0	*External Interrupt Request* 0 *Enable* (diberi nilai 1 untuk mengaktifkan)
INT2	*External Interrupt Request* 2 *Enable* (diberi nilai 1 untuk mengaktifkan)

Berikut contoh program penggunaan interupsi pada ATmega8535.

```
//INTERUPSI EKSTERNAL 0  DAN EKSTERNAL 1
#include<mega8535.h>
void main()
{ DDRC=0xff;
PORTC=0xff;
DDRD=0xff;
PORTD=0xff;
GICR=0b11000000;
#asm("sei")
while(1)
{;}}
 interrupt [2] void interupsi_ext0(void)
 { PORTC=0xF0; }
 interrupt [3] void interupsi_ext1(void)
 { PORTC=0x0F;}
```

2.10.7 Komunikasi Serial ATmega8535

ATmega8535 telah dilengkapi *Universal Synchronous and Asynchronous serial Receiver and Transmitter* (USART). Terdapat 2 jenis cara komunikasi USART, yaitu : sinkron dan asinkron. Melalui USART dapat dilakukan komunikasi serial dengan komputer atau perangkat lainnya.

Komunikasi USART dilakukan melalui *kaki* RXD (PD0) dan TXD (PD1) serta *kaki* XCK (PB0) untuk komunikasi serial sinkron. Untuk mengontrol USART digunakan *register* UCSRA, UCSRB dan UCSRC (USART *Control and Status Register* A,B,C) dan UBRR (USART *Baud Rate Register*). Untuk mengirim dan terima digunakan register UDR (USART *Data Register*), menulis ke UDR akan mengirimkan data sedangkan membaca dari UDR akan mengambil data dari *buffer*.

Tabel 2.16 UCSRA (USART *Control and Status Register* A)

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

Keterangan :

90. RXC (USART *Receive Complete*). *Bit* ini akan aktif jika ada data masuk.
91. TCX (USART *Transmit Complete*). *Bit* ini akan aktif pada saat selesai melakukan pengirim data.
92. UDRE (USART *Data Register Empty*). *Bit* ini menandakan transmitter siap untuk menerima data.
93. FE (*Framing Error*). *Bit* ini akan aktif jika terdapat *error* saat menerima data.
94. DOR (*Data OveRun*). *Bit* ini akan aktf jika ada data yang masuk namun register UDR penuh (belum dibaca).
95. PE (*Parity Error*). *Bit* ini akan aktif jika terdapat *error parity* saat menerima data.

96. U2X (*Double* USART *Transmission Speed*). *Bit* ini digunakan untuk menganda kecepatan *baud rate*.
97. MPCM (*Multi-processor Communication Mode*). *Enable Multi-processor Communication.*

Tabel 2.17 UCSRB (USART *Control and Status Register* B)

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

Keterangan :

98. RXCIE (USART *Receive Complete Interrupt Enable*)
99. TCX (USART *Transmit Complete Interrupt Enable*)
100. UDRIE (USART *Data Register Empty Interrupt Enable*)
101. RXEN (*Receive Enable*)
102. TXEN (*Transmit Enable*)
103. UCSZ2 (*Character Size*)
104. RXB8 (*Receive Data Bit* 8). Menampung *bit* ke 8 pada penerimaan 9-*bit*.
105. TXB8 (*Transmit Data Bit* 8).

Tabel 2.18 UCSRC (USART *Control and Status Register* C)

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

Keterangan :

106. URSEL (*Register Select*). Karena UCSRC memiliki alamat yang sama dengan UBBRH *bit* ini harus aktif untuk menulis ke UCSRC.

107. UMSEL (USART *Mode Select*). Memilih *mode Asynchronous* (0) atau *Synchronous* (1)

108. UPM1:0 (*Parity Mode*).

109. USBS (*Stop Bit Select*). Memilih ukuran *stop bit* (0 = 1-*bit*, 1 = 2-*bit*).

110. UCPOL (*Clock Polarity*). Mengatur pemicuan pengiriman dan penerimaan data pada *mode synchronous*.

Berikut contoh program penggunaan komunikasi serial antara ATmega8535 dengan komputer. Proram berikut menerima data dari komputer lalu mengirimkannya kembali ke komputer :

```
#include <mega8535.h>
#include <stdio.h>
void main(void)
{
char c;
UCSRA=0x00;
UCSRB=0x98;
```

```
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;
while (1)
    { c=getchar(); putchar(c);     };}
```

Saluran serial pada mikrokontroler terdiri atas dua kaki yaitu RXD dan TXD, RXD berfungsi untuk menerima data dari komputer/perangkat lainnya, TXD berfungsi untuk mengirim data ke komputer atau perangkat lainnya.

Saluran serial pada komputer atau disebut saluran RS-232 memberikan kemudahan untuk komunikasi antara komputer dengan perangkat lainnya. Tidak seperti saluran paralel yang memiliki tingkat tegangan logika (0 – 5 *Volt*), saluran serial memiliki tegangan positf dan negatif.

Standar komunikasi serial untuk komputer ialah RS-232, RS-232 mempunyai standar tegangan yang berbeda dengan saluran serial mikrokontroler, sehingga agar sesuai dengan RS-232 maka dibutuhkan suatu rangkaian *level conditioner*.

2.11 Pemrograman Bahasa C ATmega8535

2.11.1 Bentuk Dasar

Sebuah program dalam bahasa C setidaknya harus memiliki sebuah fungsi. Fungsi dasar ini disebut fungsi utama (fungsi main) dan memiliki kerangka program sebagai berikut:

```
void main (void)
{
//pernyataan-pernyataan
}
```

Jika kita memiliki beberapa fungsi yang lain, maka fungsi utama inilah yang memiliki kedudukan paling tinggi dibandingkan fungsi-fungsi yang lain, sehingga setiap kali program dijalankan akan selalu dimulai dari memanggil fungsi utama terlebih dahulu. Fungsi-fungsi yang lain dapat dipangil setelah fungsi utama dijalankan melalui pernyataan-pernyataan yang berada di dalam fungsi utama.

Contoh:

```
void inisialisasi_port (charA, charB, charC, charD); //prototipe fungsi inisialisasi
port
{ DDRB = B; DDRC = C; DDRD = D;}
void main (void) //fungsi utama
{Inisialisasi_port (0xFF, 0xF0, 0x0F, 0x00); } ;
```

2.11.2 Pengenal

Pengenal atau *identifier* merupakan sebuah nama yang didefinisikan oleh pemrogram untuk menunjukkan identitas dari sebuah konstanta, variabel, fungsi, label atau *type* data khusus. Pemberian nama sebuah pengenal dapat ditentukan bebas sesuai keinginan pemrogram tetapi tetap mengikuti aturan berikut ini:

111. Karakter pertama tidak boleh menggunakan angka.
112. Karakter kedua dapat berupa huruf, angka atau garis bawah.
113. Tidak boleh menggunakan spasi.
114. Bersifat *case sensitive*, artinya huruf kapital dan huruf kecil dianggap berbeda.
115. Tidak boleh menggunakan kata-kata yang merupakan sintaks maupun operator dalam pemrograman C, misalnya: *void, bit, long, char* dan lain-lain.

2.11.3 Konstanta dan Variabel

Konstanta berisi data yang nilainya tetap, tidak dapat diubah selama program dijalankan.

const [tipe_data] [nama_konstanta]=[nilai]
const char kontantaku = 0x10;

Variabel berisi data yang bisa berubah nilainya pada saat program dijalankan.

[tipe_data] [nama_variabel]=[nilai_awal]
char variabelku; //atau char variabelku = 0x20;

Tabel 2.19 Tipe Data, Ukuran dan Jangkauan Nilai

Tipe Data	Ukuran	Jangkauan Nilai
Bit	1 *bit*	0 atau 1
Char	1 *byte*	-128 s/d 127
Unsigned Char	1 *byte*	0 s/d 255
Signed Char	1 *byte*	-128 s/d 127
Int	2 *byte*	-32.768 s/d 32.767
Short Int	2 *byte*	-32.768 s/d 32.767
Unsigned Int	2 *byte*	0 s/d 65.535
Signed Int	2 *byte*	-32.768 s/d 32.767
Long Int	4 *byte*	-2.147.483.648 s/d 2.147.483.647
Unsigned Long Int	4 *byte*	0 s/d 4.294.967.295
Signed Long Int	4 *byte*	-2.147.483.648 s/d 2.147.483.647
Float	4 *byte*	1.2*10-38 s/d 3.4*1038
Double	4 *byte*	1.2*10-38 s/d 3.4*1038

2.11.4 Fungsi Pustaka

Bahasa C memiliki sejumlah fungsi pustaka yang berada pada *file-file* tertentu dan sengaja disediakan untuk menangani berbagai hal dengan cara memanggil

fungsi-fungsi yang telah dideklarasikan di dalam *file* tersebut. Dalam banyak hal, pustaka yang tersedia tidak berbentuk kode sumber, melainkan dalam bentuk yang telah dikompilasi. Pada saat proses *linking*, kode-kode dari fungsi ini akan dikaitkan dengan kode-kode yang dituliskan oleh pemrogram. Sintaks untuk menggunakan fungsi pustaka adalah sebagai berikut:

Contoh:

```
# include<lcd.h>
# include<string.h>
# include<mem.h>
```

2.11.5 Pengarah Prosesor

Digunakan untuk mendefinisikan prosesor yang digunakan, dalam hal ini adalah untuk mendefinisikan jenis mikrokontroler yang digunakan. Dengan pengarah preprosesor ini, pendeklarasian *register-register* dan penamaannya dilakukan pada *file* lain yang disisipkan dalam program utama dengan sintaks sebagai berikut:

```
# include<nama_prosesor>
# include<mega8535.h> // Contoh
```

2.11.6 Operator Aritmatika

Operator aritmatika merupakan beberapa operator yang digunakan untuk melakukan perhitungan aritmatika.

Tabel 2.20 Operator Aritmatika

Operator	Keterangan
+	Penjumlahan

-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa Pembagian

2.11.7 Operator *Bit*

Operator *bit* merupakan operator logika yang bekerja pada tingkat bit. Berbeda dengan operator logika, dimana pada operator logika menghasilkan benar atau salah (*boolean*), pada operator ini akan dihasilkan data dalam bentuk biner.

Tabel 2.21 Operator *bit*

Operator	Keterangan
&	Operasi *AND level bit* (biner)
|	Operasi OR
^	Operasi XOR
~	Operasi NOT
<<	Operasi untuk menggeser ke kiri
>>	Operasi untuk mengeser ke kanan

a = 0x12;
b = 0x34;

maka, a&b = 0x10 dan a<<1 = 0x24

2.11.8 Operator Penugasan

Operator penugasan digunakan untuk memberikan nilai atau memanipulasi data sebuah variabel.

Tabel 2.22 Operator Penugasan

Operator	Keterangan
=	Operator untuk memberi nilai variabel
+=	Operator untuk menambahkan nilai variabel
-=	Operator untuk mengurangi nilai variabel

*=	Operator untuk mengalikan nilai variabel
/=	Operator untuk membagi nilai variabel
%=	Oprator untuk memperoleh sisa pembagian

Contoh:

a +=2;
*/*artinya nilai variabel a berubah menjadi a=a+2b *=4; artinya nilai variabel b berubah menjadi b=b*2 */*

Selain operator penugasan di atas, juga ada operator penugasan yang berkaitan dengan operasi *bit* seperti tabel berikut:

Tabel 2.23 Operator Penugasan *Bit*

Operator	Contoh	Arti
&=	X&=1	Variabel x di-AND-kan dengan 1
|=	x|=1	Variabel x di-OR-kan dengan 1
^=	x^=1	Variabel x di_XOR-kan dengan 1
<<=	x<<1	Variabel x digeser ke kiri 1 kali
>>=	x>>1	Variabel x digeser ke kanan 1 kali

2.11.9 Operator Penambahan dan Pengurangan

Merupakan operator yang digunakan untuk menaikkan atau menurunkan nilai sebuah variabel dengan selisih 1.

Tabel 2.24 Operator Penambahan dan Pengurangan

Operator	Keterangan
++	Penambahan nilai variabel dengan 1
--	Pengurangan nilai variabel dengan 1

x++ ; //sama saja seperti x=x+1
x--; // sama saja seperti x=x-1

2.11.10 Pernyataan *If...Else*

Hampir sama dengan pernyataan *if*, yaitu digunakan untuk melakukan keputusan terhadap dua buah kemungkinan, yang membedakan adalah kedua kemungkinan yang akan diputuskan masing-masing mengerjakan suatu blok pernyataan atau mengerjakan blok pernyataan yang lain.

```
If(kondisi )
{//blok pernyataan yang akan dikerjakan jika kondisi if dipenuhi
}
Else
{//blok pernyataan yang akan dikerjakan jika kondisi if tidak dipenuhi
}
```

Contoh:

```
If(PINA>0x 80)
{Dataku = PINA;
PORTC=0xFF; }
Else
{Dataku = ~PINA;
PORTC=0x00;}
```

Pernyataan *if...else* di atas memeriksa apakah data yang terbaca oleh Port A (PINA) nilainya lebih dari 0x80 atau tidak, jika "ya", maka variabeldataku diisi dengan nilai PINA dan data 0xFF dikeluarkan ke PORTC, tetapi jika tidak, maka variabel dataku diisi dengan nilai komplemen dari PINA dan data 0x00 dikeluarkan ke PORTC.

2.11.11 Pernyataan *Switch*

Digunakan untuk melakukan pengambilan keputusan terhadap banyak kemungkinan.

```
Switch(ekspresi)
{
Case nilai_1 : pernyataan_1;break;
Case nilai_2 : pernyataan_2;break;
Case nilai_3 : pernyataan_3;break;
...
Default : pernyataan_default ; break ;
}
```

Pada pernyataan switch, masing-masing pernyataan (pernyataan_1 sampai dengan pernyataan *default*) dapat berupa satu atau beberapa perintah dan tidak perlu berupa blok pernyataan. Pernyataan_1 akan dikerjakan jikaekspresi bernilai sama dengan nilai_1, pernyataan_2 akan dikerjakan jika ekspresi bernilai sama dengan nilai_2 dan seterusnya. Pernyataan *default* bersifat optional, artinya boleh ada boleh tidak. Setiap akhir dari pernyaan harus diakhiri dengan *break*, karena digunakan untuk keluar dari pernyataan *switch*.

Contoh:

```
Switch(PIN A)
{
Case 0xFE : PORTC=0x00;break;
Case 0xFD : PORTC=0xFF;break;
}
```

Artinya, PORTA akan dibaca dan dicocokkan datanya (PINA) dengan nilai *case*.

2.11.12 Pernyataan *While*

Digunakan untuk pengulangan sebuah pernyataan atau blok pernyataan secara terus menerus selama kondisi tertentu masih terpenuhi. Bentuk pernyataan *while* adalah sebagai berikut:

```
While(kondisi )
{// sebuah pernyataan atau blok pernyataan }
```

Jika pernyataan yang akan diulang hanya berupa sebuah pernyataan saja maka tanda {dan} dapat dihilangkan.

Contoh:

```
unsigned char a=0;
...
while(a<10)
{PORTC=a; a++; }
```

Pernyataan di atas akan mengeluarkan data a ke PORTC secara berulang-ulang. Setiap kali pengulangan, nilai a akan bertambah 1 dan setelah nilai a tidak lagi lebih kecil dari 10 maka pengulangan akan berhenti.

2.11.13 Pernyataan *For*

Digunakan untuk melakukan pengulangan sebuah pernyataan atau blok pernyataan, tetapi berapa kali jumlah pengulangannya dapat ditentukan terlebih spesifik.

```
for (nilai_awal , kondisi , perubahan)
{
//sebuah pernyataan atau blok pernyataan
}
```

Contoh:

```
Unsigned int a;
For (a=1 , a<10, a++)
{PORTC=a; }
```

Pertama kali nilai a adalah 1, kemudian data a dikeluarkan ke PORTC. Selanjutnya, data a dinaikkan (a++) jika kondisi a < 10 masih terpenuhi, maka data a akan terus dikeluarkan ke PORTC.

2.11.14 *Array*

Array merupakan sekumpulan data dengan tipe yang sama yang dideklarasikan dalam satu nama variabel. *Array* dapat memiliki dimensi satu, dua atau lebih, tetapi umumnya hanya sampai 3 dimensi.

```
Char data[ 7] ;
```

Artinya, variabel *array* dengan namadata terdiri dari 8 elemen data yang bertipe sama, yaitu : *char*.

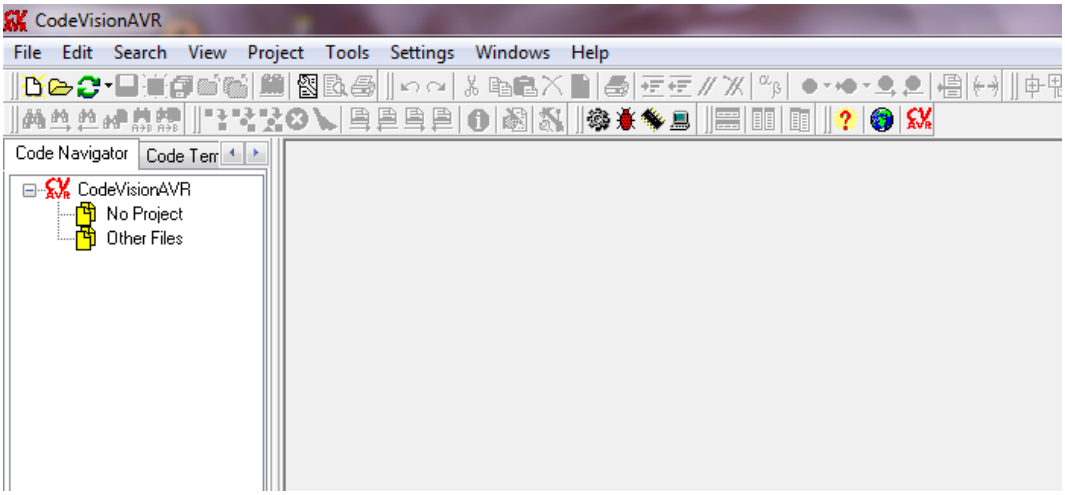
Nama_array [indeks]; Indeks adalah penunjuk data elemen tertentu pada variabel array yang nilainya selalu dimulai dari 0.

```
char dataku[7] = {0, 1, 2, 3, 4, 5, 6};
char data[]; //tidak boleh dilakukan
char data[]={0, 1, 2, 3, 4, 5, 6};
char data[]={"Mikrokontroler ATmega8535"}; //atau Char
data[]="Mikrokontroler ATmega8535";
```

2.11.15 CodevisionAVR

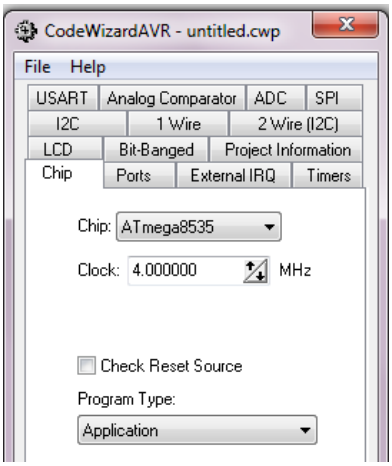
CodevisionAVR merupakan *compiler* bahasa C, didesain khusus untuk mikrokontroler keluarga Atmel AVR. Dapat digunakan pada Windows 98, Me, NT 4, XP, dan Vista. Fungsi utama CodevisionAVR ialah untuk menuliskan program yang akan kita masukkan kedalam mikrokontroler. Selain dengan CodevisionAVR masih ada jenis lainnya (*compiler* lain) yakni WinAVR.

Selain library standar C, CodeVisionAVR C compiler juga memiliki library lainnya, yaitu *alphanumeric* LCD *modules*, *philips* I2C bus, *national semiconductor* LM75 *temperature sensor*, dan sebagainya.



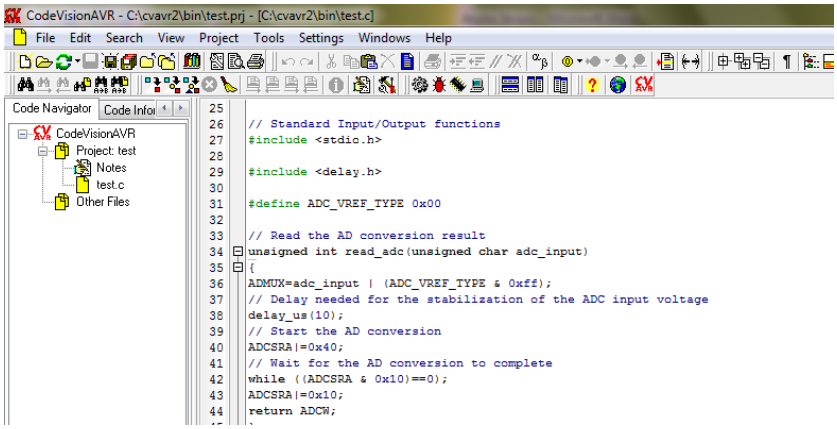
Gambar 2.28 Tampilan Awal CodeVisionAVR

CodeVisionAVR memiliki CodeWizardAVR yang memudahkan dalam penulisan program inisialisasi AVR. Dengan cara mengatur konfigurasi dari *Chip*, *PORT*, ADC, USART, *Timer* dan sebagainya.



Gambar 2.29Tampilan CodeWizardAVR

Setelah mengatur konfigurasi inisialisasi yang akan digunakan, maka secara otomatis program inisilisasi telah tertulis pada CodeVisionAVR.

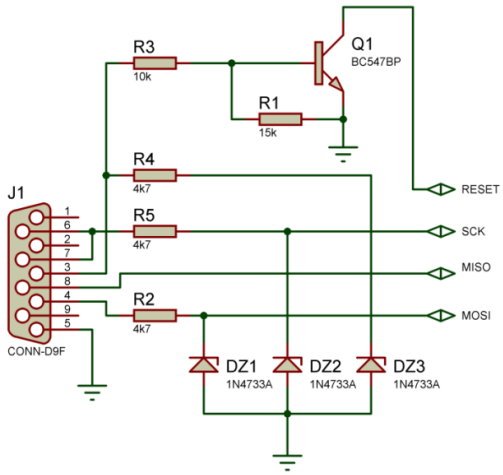


Gambar 2.30 Program inisialisasi dari konfigurasi CodeVisionAVR

Agar program dalam bahasa C dapat ditulis ke dalam ATmega8535, maka terlebih dahulu diubah menjadi *file* yang berekstensi ".hex". CodeVisionAVR telah menyediakan file yang berekstensi ".hex", walaupun program yang dirancang adalah program yang menggunakan bahasa C.

2.11.16 Pengunduh Program ATmega8535

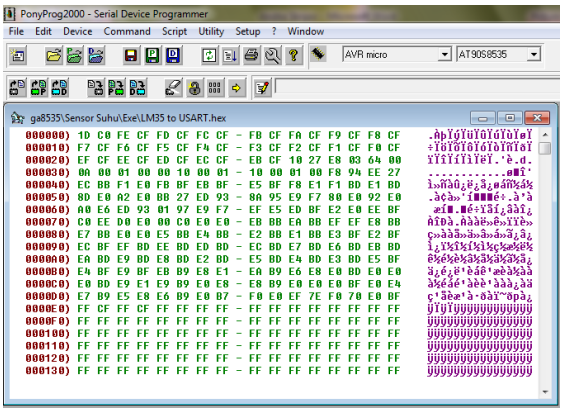
Ponyprog merupakan perangkat lunak yang dapat mengunduhkan program kedalam AVR. Ponyprog mendukung pengunduh menggunakan pengunduh serial. Program yang diunduhkan berupa *file* yang berekstensi ".hex". Ponyprog mempunyai *default* rangkaian untuk mengunduh ke dalam AVR.



Gambar 2.31 Rangkaian Pengunduh Serial Ponyprog

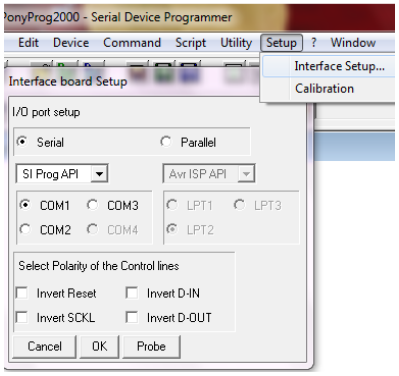
Konektor DB9 pada rangkaian diatas terhubung dengan saluran serial DB9 pada komputer. Kemudian keluaran rangkaian pengunduh serial ponyprog terubung dengan pin reset, SCK, MISO dan MOSI pada AVR. Berikut langkah-langkah pengunduhan AVR menggunakan pengunduh serial ponyprog :

1. Hal pertama yang perlu dilakukan adalah jalankan ponyprog dan kemudian mengambil bahasa C yang telah berekstensi .hex.



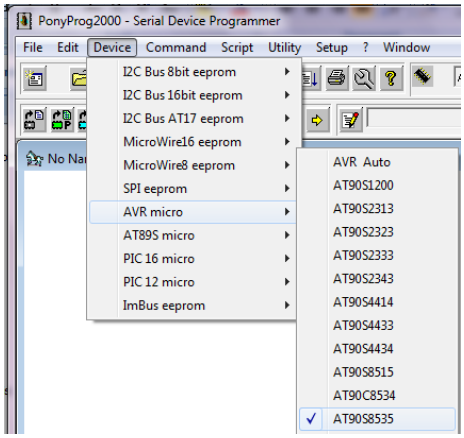
Gambar 2.32 Tampilan Ponyprog

2. Menkonfigurasi interface *board setup,* yakni menentukan alamat saluran serial (COM1,COM2 atau COM3).



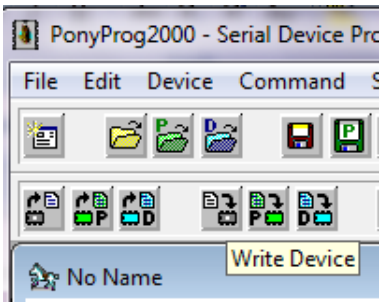
Gambar 2.33 Konfigurasi Antarmuka Pengunduh Serial Ponyprog

3. Menentukan AVR yang digunakan



Gambar 2.34 Konfigurasi AVR yang Digunakan

4. Mulai memindahkan program ke AVR dengan menekan *write device*.



Gambar 2.35 Menulis Program ke AVR