BAB IV

PEMBAHASAN

4.1 Proses Kerja Sistem Pencacah Nuklir

Sistem Pencacah Nuklir adalah sebuah alat yang digunakan untuk mencacah intensitas radiasi yang ditangkap oleh detektor nuklir dalam selang waktu tertentu. Dimana alat tersebut dapat dioperasikan melalui komputer serta dapat mengirimkan hasil pencacahan ke komputer maupun ke dalam *lcd display*. Dalam hal ini terdapat beberapa komponen penting yang perlu diperhatikan, yaitu data yang diterima dari komputer, data yang ditampilkan ke dalam *lcd display*, aktivasi *timer* dan *counter*, data yang dikirim ke komputer. Masing - masing penjelasannya akan diuraikan sebagai berikut :

4.1.1 Data yang Diterima dari Komputer

Data yang diterima adalah informasi mengenai jumlah dan rentang waktu pencacahan yang berupa tipe data string dengan panjang data sebanyak 6 karakter yang mewakili 3 buah informasi data. Masing – masing informasi data tersebut adalah 2 karakter awal mewakili informasi jumlah proses pencacahan, 2 karakter selanjutnya mewakili informasi waktu pencacahan dalam satuan menit dan 2 karakter terakhir mewakili informasi waktu pencacahan dalam satuan detik.

4.1.2 Data yang Ditampilkan ke Dalam *LCD Display*

Informasi yang ditampilkan oleh *lcd display* pada saat alat baru menyala adalah informasi untuk memberikan perintah aktivasi pencacahan melalui komputer. Setelah *microcontroller* mendapatkan kiriman data dari

komputer maka tampilan akan berubah dengan tiap – tiap baris yang mengandung informasi tertentu, yaitu :

- Baris pertama berupa keterangan
- Baris kedua berupa informasi tentang hasil cacah persatuan waktu dalam satu kali proses pencacahan.
- Baris ketiga berupa informasi tentang rentang waktu dalam satu kali proses pencacahan. Ditunjukkan dalam menit dan detik.
- Baris keempat berupa informasi tentang proses pencacahan keberapa dari jumlah pengulangan pencacahan yang akan dilakukan.

4.1.3 Aktivasi *Timer* dan *Counter*

Setiap mendapatkan kiriman informasi dari komputer maka *microcontroller* segera mengaktifkan *timer* dan *counter* untuk memulai proses pencacahan. Pada *timer* yang digunakan sebagai pewaktu, proses pencacahan dibatasi sesuai dengan informasi data waktu yang telah diterima sebelumnya dari komputer. Pada saat ini *counter* sedang melakukan pencacahan terhadap masukan dari sensor. Setelah batas waktu telah tercapai maka secara bersamaan *timer* dan *counter* di non-aktifkan. *Timer* dan *counter* akan segera aktif kembali hingga jumlah proses pencacahan sama dengan jumlah proses pencacahan yang telah diterima sebelumnya dari kiriman komputer.

4.1.4 Data yang Dikirim ke Komputer

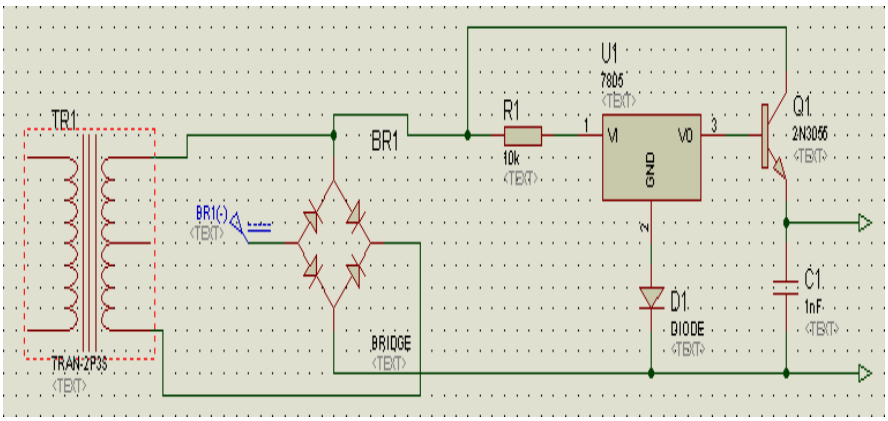
Pengiriman hasil dari proses pencacahan di kirim ke komputer setiap kali sebuah proses pencacahan telah dilakukan. Data yang dikirim berupa formasi jumlah pencacahan.

Gambar 4.1 Blok Diagram Proses Sistem Pencacah Nuklir

4.2 Perancangan Rangkaian Elektronika *GM Counter*

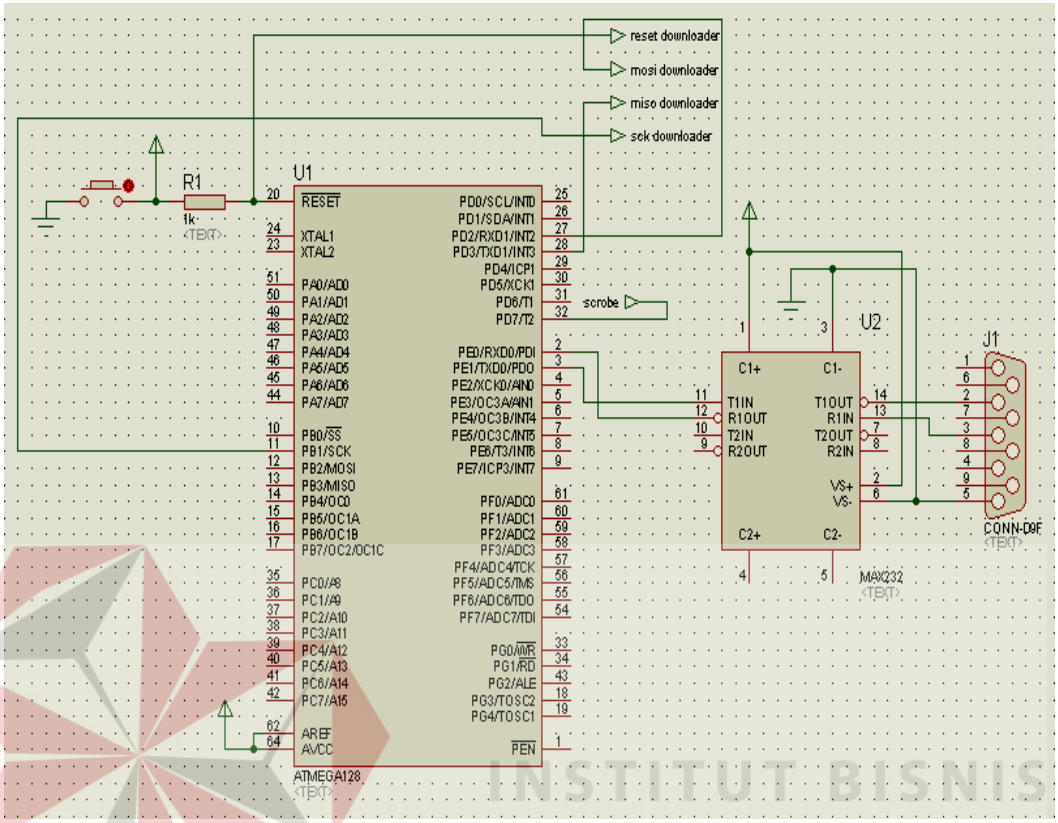
Berikut ini adalah *schematic* dari rangkaian elektronika *GM Counter* :

1. Rangkaian *Adaptor*



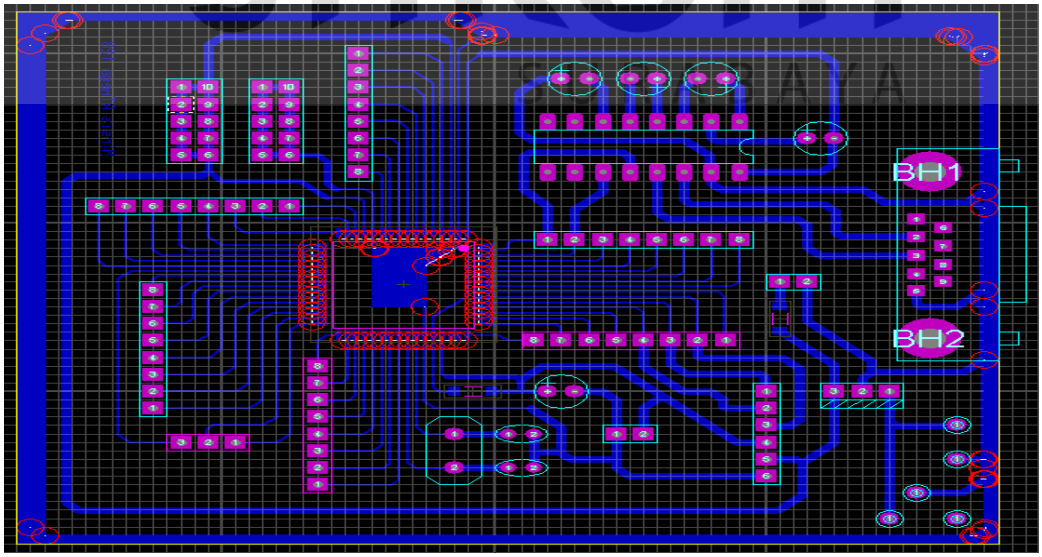
Gambar 4.2 *Schematic Adaptor*

2. Rangkaian *Minimum System*



Gambar 4.3 *Shematic Minimum System*

3. Layout *GM Counter*



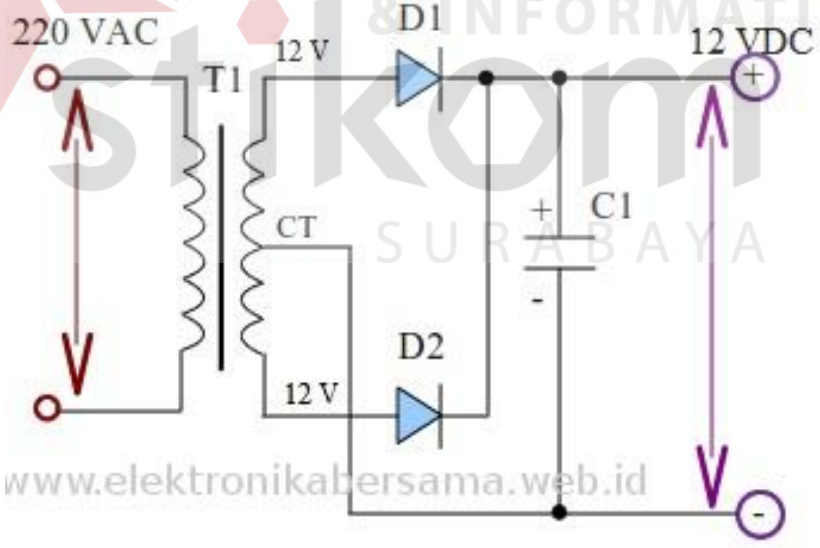
Gambar 4.4 *Layout GM Counter*

4.2.1 Rangkaian *Adaptor*

Agar alat dapat dioperasikan dengan menggunakan sumber daya dari PLN yang merupakan tegangan AC dengan tegangan sekitar 240v maka diperlukan rangkaian adaptor sebagai konversi tegangan AC menjadi tegangan DC serta menurunkan tegangannya sesuai dengan kebutuhan dari rangkaian alat *GM Counter* yang sebesar kurang lebih 5v. Berikut ini adalah komponen utama pada rangkaian *Adaptor :*

1. ***Transformator CT***

   Berfungsi untuk menurunkan tegangan tinggi AC (Bolak-balik) 240v atau 220v menjadi tegangan rendah sebesar 12v. Selain itu dengan ditambahkanya 2 buah dioda membuat arus yang dihasilkan menjadi searah DC.



Gambar 4.5 Rangkaian *Transformator CT*

2. 7805

Digunakan untuk meregulasi tegangan yang keluar dari *transformator CT* yang sebesar 12v menjadi 5v.(7)

Gambar 4.6 Komponen 7805

3. *Tansistor 2N3055*

Arus yang masuk ke dalam *7805* memiliki besaran sekitar 0.6 ampere, hal ini mengakibatkan *7805* mengalami panas yang berlebihan. Untuk itu dibutuhkan *Transistor 2N3055* yang merupakan jenis transistor NPN agar arus yang melewati *7805* tidak melebihi kapasitas, sehingga arus yang sebesar 0.6 ampere mengalir melalui *Transistor* sedangkan arus yang mengalir pada *7805* yang sebesar 0.5 ampere (sesuai dengan spesifikasi 7805) berfungsi sebagai *trigger* pada transistor *2N3055*.(6)

Gambar 4.7 Komponen 2N3055

4.2.2 Rangkaian *Minimum System*

Agar *microcontroller* dapat berfungsi dan bekerja dengan semestinya maka maka diperlukan adanya rangkaian *Minimum System.* Berikut ini adalah komponen yang ada di dalam *Minimum System* :

1. *Reset*

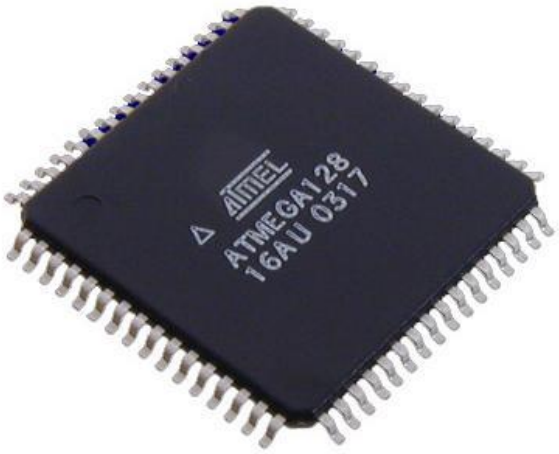
Berfungsi untuk mereset *microcontroller* sehingga mendapatkan kondisi *microcontroller* seperti semula kembali. Penggunaan reset dilakukan dengan menghubungkan ground dengan pin 20 pada *microcontroller* melalui sebuah *push button* dengan rangkaian *pull up* , hal tersebut dikarenakan reset *microcontroller* bersifat aktif *low*.

2. *ISP (In System Programming)*

Penggunaa *downloader* dilakukan dengan cara berikut :

- *MOSI* pada *downloader* dihubungkan dengan pin 27 atau *RXD1 microcontroller.*
- *MISO* pada *downloader* dihubungkan dengan pin 28 atau *TXD1 microcontroller.*
- *SCK* pada *downloader* dihubungkan dengan pin 11 atau *SCK microcontroller.*
- *RESET* pada *downloader* dihubungkan dengan pin 20 atau *RESET microcontroller.*
- *VCC* pada *downloader* dihubungkan dengan catu daya sebesar 5v.
- *Ground* pada *downloader* dihubungkan dengan *ground.*

3. *Microcontroller ATMEGA 128*



Gambar 4.8 ATMEGA 128

Secara *default* nilai *internal RC Oscillator clock ATMEGA 128* adalah sebesar 1MHz. Untuk mengoptimalkan kemampuan *microcontroller* sebagai alat pencacah *GM Counter* maka nilai clock *internal RC Oscillator* ditingkatkan sebesar 8MHz. Untuk menaikkan nilai clock adalah dengan mengubah nilai *fuse bit ATMEGA 128* menggunakan sebuah program *Khazama AVR*. Hal yang perlu diperhatikan dalam mengubah nilai *fuses bit* agar *ATMEGA 128* dapat bekerja sesuai dengan keinginan adalah sebagai berikut :

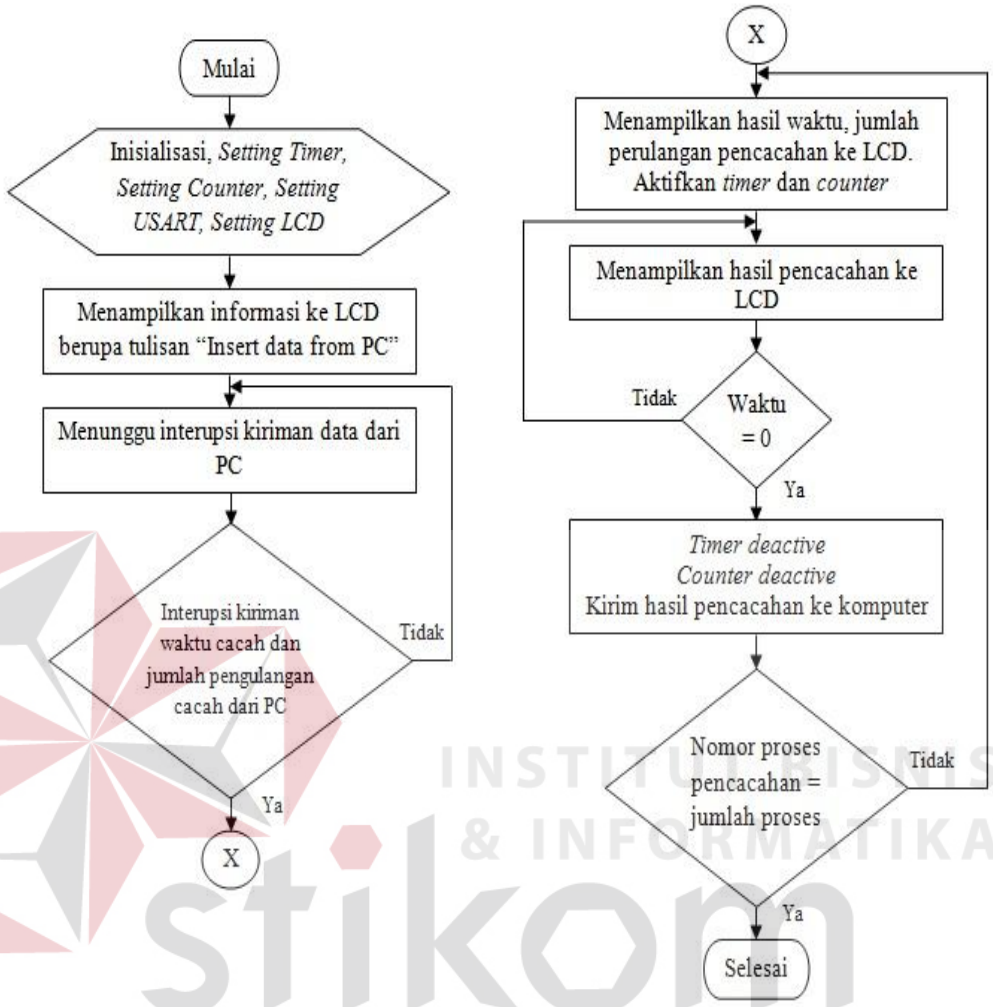
- SUT1 dan SUT0 diset secara berurutan 0 dan 1.

  Hal ini dilakukan dengan tujuan agar memperoleh mode *Fast Rising Power.*

- CKSEL3 ~ 0 diset secara berurutan 0,1,0 dan 0.

  Hal ini dilakukan agar nilai clock yang digunakan bernilai 8MHz.

4.3 Perancangan Progam *GM Counter*



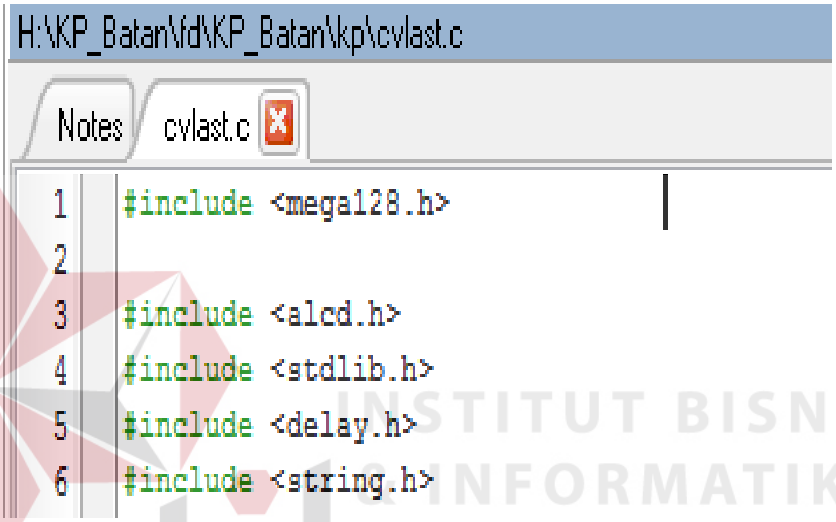
Gambar 4.9 *Flowchart* Program *GM Counter*

4.3.1 *Code Vision AVR*

Merupakan salah satu pemrograman *microcontroller AVR.* Pemrograman *Code Vision* menggunakan bahasa C sehingga mudah untuk digunakan. Berikut adalah pemrograman yang akan di-*load* ke dalam *microcontroller* :

1. *Library*

Sebuah fitur yang istilahnya umum di dalam dunia pemrograman. Fungsi bawaan ini telah disediakan agar mempermudah dalam memprogram, yang mana ini merupakan suatu keistimewaan dari sebuah program dalam membuat perintah – perintah khusus yang dapat langsung dipanggil dalam program.



Gambar 4.10 *Source code* library

2. *Initialisasi*

Merupakan pengenalan suatu variabel baik berupa tipe data yang digunkan maupun nilai awal dari variabel tersebut di dalam pemrograman.

H:\KP_Batan\fd\KP_Batan\kp\cvlast.c

Notes | cvlast.c

```
7   #ifndef RXB8
8   #define RXB8 1
9   #endif
10
11  #ifndef TXB8
12  #define TXB8 0
13  #endif
14
15  #ifndef UPE
16  #define UPE 2
17  #endif
18
19  #ifndef DOR
20  #define DOR 3
21  #endif
22
23  #ifndef FE
24  #define FE 4
25  #endif
26
27  #ifndef UDRE
28  #define UDRE 5
29  #endif
30
31  #ifndef RXC
32  #define RXC 7
33  #endif
34
35  #define FRAMING_ERROR (1<<FE)
36  #define PARITY_ERROR (1<<UPE)
37  #define DATA_OVERRUN (1<<DOR)
38  #define DATA_REGISTER_EMPTY (1<<UDRE)
39  #define RX_COMPLETE (1<<RXC)
40
41  unsigned char jum[5], men[5], det[5], nom[1], buff[5];
42  unsigned int rentang, nomor ,waktu = 1, jumlah ,menit ,detik, i = 0, j;
43  unsigned long int tot, cacah;
```

Gambar 4.11 *Source code* initialisasi untuk komunikasi *USART* dan variabel – variabel yang dibutuhkan

```
46  #define RX_BUFFER_SIZE0 8
47  char rx_buffer0[RX_BUFFER_SIZE0];
48
49  #if RX_BUFFER_SIZE0 <= 256
50  unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
51  #else
52  unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
53  #endif
```

Gambar 4.12 *Source code* initialisasi untuk penerimaan di dalam *USART*

4.3.1 Komunikasi dengan komputer

Agar dapat menerima data dari komputer dan mengirim data ke komputer diperlukan fitur *USART* dalam pemrograman. Untuk menggunakan fungsi *USART0* sesuai dengan yang diinginkan maka perlu terlebih dahulu merubah nilai register di dalamnya. Berikut adalah regiter yang digunakan pada *USART GM Counter*:

- UDR = terdapat 8bit yang digunakan sebagai tempat penampungan sementara (*buffer)* saat mengirim dan menerima data.
- UCSR0B = diset dengan nilai D8 hexa. Sehingga fungsi –fungsi yang digunakan adalah :
  a. Bit 7(RXCIE) digunakan untuk mengaktifkan fungsi interupsi dalam penerimaan data.
  b. Bit 6(TXCIE) digunakan untuk mengaktifkan fungsi interupsi dalam pengiriman data.
  c. Bit 4(RXEN) digunakan agar *USART* dapat menerima data.
  d. Bit 3(TXEN) digunakan agar *USART* dapat mengirim data.
- UCSR0C = diset dengan nilai 6hexa. Sehingga mode yang digunakan adalah :
  a. *USART mode select* = operasi *USART* menggunakan mode *Asynchronous*
  b. *Parity mode* = tidak digunakan atau di non-aktifkan.
  c. *Stop bit select* = menggunakan *stop bit* dengan jumlah 1bit.

d. *Character size* = ukuran karakter berjumlah 8bit(karena UCSZ2 pada UCSRB bernilai 0).

e. *Clock polarity* = *Rising XCK(synchronous clock)* pada pengiriman data dan *falling XCK(synchronous clock)* pada penerimaan data.

- UBRR0L = diset dengan nilai 33hexa dengan tujuan mendapatkan *baud rate* sebesar 9600.

Tabel 4.1 Hasil eror yang dihasilkan nilai *Baud rate* pada setingan UBRs

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5Mbps		1Mbps		691.2Kbps		1.3824Mbps		921.6Kbps		1.8432Mbps	

1. UBRR = 0, Error = 0.0%

1. Penerimaan data dari komputer

Penerimaan data menggunakan fitur interupsi penerimaan data pada *USART0*. Dengan demikian setiap kedatangan per karater data yang ditampung ke dalam *UDR* segera disimpan ke dalam variabel *array*. Setelah data yang diterima telah terkumpul maka akan dilanjutkan dengan proses pembacaan data atau pemisahan data.

```
93  interrupt [USART0_RXC] void usart0_rx_isr(void)
94  {
95  char status,data;
96  status=UCSR0A;
97  data=UDR0;
98  if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
99     {
100    rx_buffer0[rx_wr_index0++]=data;
101    buff[i]=data;
102    if(i<5)
103     i++;
104    else
105     init();
106
107 #if RX_BUFFER_SIZE0 == 256
108    // special case for receiver buffer size=256
109    if (++rx_counter0 == 0)
110       {
111 #else
112    if (rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
113    if (++rx_counter0 == RX_BUFFER_SIZE0)
114       {
115       rx_counter0=0;
116 #endif
117       rx_buffer_overflow0=1;
118       }
119    }
120 }
121
```

Gambar 4.13 *Source code* penerimaan data dari komputer

```
61  for(j=0;j<6;j++)
62      {
63          if(j<2)
64              jum[j]=buff[j];
65          else if(j<4)
66              men[j-2]=buff[j];
67          else
68              det[j-4]=buff[j];
69      };
70  jumlah = atoi(jum);
71  menit = atoi(men);
72  detik = atoi(det);
73  rentang=detik+(menit*60);
```

Gambar 4.14 *Source code* proses pemisahan data

2. Pengiriman data ke komputer

Pengiriman data menggunakan fungsi putchar dengan pengiriman data per karakter dan diakhiri dengan karakter "x" sebagai penanda pengiriman telah berkahir.

```
318             itoa(tot,tot_lcd);
319             for(j=0;j<=strlen(tot_lcd);j++)
320                 {putchar(tot_lcd[j]);}
321             putchar('x');
```

Gambar 4.15 *Source code* pengiriman data ke komputer

4.3.3 *Timer1*

Salah satu fitur dalam *microcontroller* untuk menghitung waktu. *Timer* yang digunakan adalah *timer1* yang memiliki ukuran 16bit. Agar *timer1* dapat berfungsi sesuai dengan yang diinginkan maka perlu merubah nilai yang terdapat di dalam register *timer*. Berikut ini adalah register yang terdapat pada *timer1* :

- TIMSK = untuk mengaktifkan *interrupt overflow* pada timer1 adalah dengan merubah bit ke 2 menjadi *high* atau 1.
- TCCR1B = Diset dengan nilai 5hexa agar timer aktif dengan prescaler 1024.
- TCNT1H = merupakan nilai *high* atau 8bit sebelah kiri dari TCNT1.
- TCNT1L = merupakan nilai low atau 8bit sebelah kanan dari TCNT1.

Untuk menentukan nilai TCNT1H dan TCNT1L maka hal – hal yang perlu diperhatikan adalah :

1. *Clock* yang digunakan adalah sebesar 8MHz sehingga untuk mencari periode adalah 1/8MHz = 0.000000125s atau 125ps
2. *Prescaller* yang digunakan adalah skala 1024 sehingga periode yang didapat di naikkan dengan skala sebesar 1024. 125ps x 1024 = 0.000128s atau 128ms.
3. Waktu yang diharapkan dalam satu siklus interupsi di *timer1* adalah 1s sedangkan periode dalam 1 tick atau 1 *counting*, maka untuk mengetahui jumlah tick dalam 1s adalah 1s/128ms=7812.5.
4. Jumlah bit pada *timer1* adalah 16bit sehingga terdapat 65535. Dengan demikian maka untuk mendapatkan nilai awal TCNT1 adalah 65535-7812.5 = 57723.5.

Dengan demikian dapat diketahui nilai TCNT1H = E1hexa dan TCNT1L = 7Bhexa. Akan tetapi secara kenyataan nilai tersebut tidaklah akurat bila digunakan dengan penghitungan waktu yang cukup lama, misal untuk penghitungan waktu lebih dari 2menit. Untuk penjelasan lebih lanjut akan

dibahas pada kesimpulan di bab selanjutnya. Sehingga dalam hal ini penulis menggunakan nilai E154 dengan perhitungan dari kalibrasi secara manual dengan hasil yang lebih akurat. Di dalam *timer1* terdapat interupsi yang terjadi apabila nilai TCNT1 telah melewati batas angka 16bitnya(65535). Hal yang demikian dikondisikan sebagai waktu 1 detik, agar *timer1* dapat berjalan kembali sesuai dengan yang diinginkan maka saat *event* interupsi tersebut dilakukan pemberian nilai awal yang telah ditetapkan sebelumnya yaitu E154.

```
79  TCNT1H=0xE1;
80  TCNT1L=0x54; => nilai awal timer 1
81  TCNT2=0x00;  => nilai awal counter 2
82  lcd_clear(); lcd_puts("counter :");
83  lcd_gotoxy(0,2); lcd_puts("Time = "); lcd_puts(men);
84  lcd_gotoxy(10,2); lcd_puts("m: "); lcd_puts(det);
85  lcd_gotoxy(15,2); lcd_puts("s");
86  lcd_gotoxy(0,3); lcd_puts("At "); lcd_puts(nom);
87  lcd_gotoxy(6,3); lcd_puts("from "); lcd_puts(jum);
88  TCCR2=0x07;  => nilai register counter 2
89  TCCR1B=0x05; => nilai register timer 1
90  }
```

Gambar 4.16 *Source code* seting *timer1* dan *counter2*

```
207 interrupt [TIM1_OVF] void timer1_ovf_isr(void)
208 {
209 // Reinitialize Timer1 value
210 TCNT1H=0xE1;
211 TCNT1L=0x54;
212 // Place your code here
213 waktu--;
214 }
```

Gambar 4.17 *Source code* interupsi *timer*

4.3.4 *Counter2*

Counter merupakan kesamaan dari *timer*, hanya saja pada *counter* tidak menggunakan frekuensi yang terdapat di dalam *microcontroller* melainkan berasal dari sumber luar atau *external*. Sumber luar ini didapat dari tegangan sinyal yang terhubung pada pin 32 atau T2. Berikut ini adalah register yang terdapat pada *counter2* :

- TTCR2 = 7hexa, yang mana ini menjadikan *counter2* mendapatkan *counting* dari luar dengan perubahan yang terjadi saat tegangan yang diterima bernilai naik(rendah ke tinggi).
- TCNT2 adalah nilai pada *counter2*. Dengan memberikan nilai 0 pada saat deklarasi *counter2*(nilai awal) maka akan didapatkan rentang *counting* sebanyak 8bit atau 256.
- TIMSK = untuk mengaktifkan *interrupt overflow* pada *counter2* adalah dengan merubah nilai bit ke 6 menjadi *high* atau 1. Interupsi terjadi ketika nilai pada TCNT2 telah melebihi kapasitas *counter2* yang sebanyak 8bit atau 256. Oleh karena itu agar dapat mempertahankan nilai yang telah dihitung dalam pencacahan di dalam *counter2* adalah dengan menyimpan data hasil cacah ke dalam sebuah variabel pada saat terjadi *interrupt overflow*.

```
217 interrupt [TIM2_OVF] void timer2_ovf_isr(void)
218 {
219 // Place your code here
220     cacah=cacah+256;
221 }
```

Gambar 4.18 *Source code* interupsi *counter*

4.3.5 Fungsi void init

Berfungsi untuk memberikan nilai awal pada semua variabel. Fungsi ini adalah semacam reset proses pencacahan. Didalamnya terdapat pemberian nilai awal, proses pemisahan data yang telah diperoleh dari komputer sebelumnya, penggunaan data telah diperoleh untuk menunjang proses pencacahan dan menampilkan informasi proses yang akan dilakukan ke dalam *LCD*.

H:\KP_Batan\fd\KP_Batan\kp\cvlast.c

Notes | cvlast.c

```
57
58  void init()
59  {
60  i=0;
61  for(j=0;j<6;j++)
62      {
63          if(j<2)
64              jum[j]=buff[j];
65          else if(j<4)
66              men[j-2]=buff[j];
67          else
68              det[j-4]=buff[j];
69      };
70  jumlah = atoi(jum);
71  menit = atoi(men);
72  detik = atoi(det);
73  rentang=detik+(menit*60);
74  nomor = 1;
75  itoa(nomor,nom);
76  tot=0;
77  waktu=rentang;
78  cacah=0;
79  TCNT1H=0xE1;
80  TCNT1L=0x54;
81  TCNT2=0x00;
82  lcd_clear(); lcd_puts("counter :");
83  lcd_gotoxy(0,2); lcd_puts("Time = "); lcd_puts(men);
84  lcd_gotoxy(10,2); lcd_puts("m: "); lcd_puts(det);
85  lcd_gotoxy(15,2); lcd_puts("s");
86  lcd_gotoxy(0,3); lcd_puts("At "); lcd_puts(nom);
87  lcd_gotoxy(6,3); lcd_puts("from "); lcd_puts(jum);
88  TCCR2=0x07;
89  TCCR1B=0x05;
90  }
```

Gambar 4.19 *Source code* nilai awal, tampilan *LCD* dan aktivasi *timer* dan *counter*

4.3.6 Tampilan ke *LCD*

Hal yang perlu diketahui untuk menampilkan data ke dalam *LCD* adalah data tersebut harus bertipe data *ascii* atau *string.* Selain itu koordinat pada *LCD* diperlukan untuk menentukan posisi awal penulisan di dalam *LCD.*

```
304   lcd_init(16); lcd_clear();
305   lcd_gotoxy(0,1); lcd_puts("  please insert");
306   lcd_gotoxy(0,2); lcd_puts("  data from pc");
```

Gambar 4.20 *Source code* tampilan awal LCD

4.3.7 Proses pencacahan

Saat pencacahan sedang berrlangsung hasil dari jumlah total pencacahan yang sedang berlangsung ditampilkan ke *LCD* secara *real time*

```
332       if(TCCR1B!=0)
333       {
334       tot=TCNT2+cacah;
335       itoa(tot,tot_lcd);
336       lcd_gotoxy(0,1); lcd_puts(tot_lcd);
337       };
```

Gambar 4.21 *Source code* tampilan hasil pencacahan ke *LCD* secara *real time*

4.3.8 Proses pencacahan berakhir

Saat nilai di dalam variabel waktu telah bernilai = 0, maka *timer1* dan *counter2* dengan segera di-*deactive* atau dimatikan. Hal ini bertujuan untuk menjaga keakuratan data pencacahan sehingga tidak ada kelebihan proses pencacahan yang akan terjadi setelah waktu yang ditentukan telah berakhir. Setelah itu hasil dari proses pencacahan langsung dikirimkan ke komputer dan

juga ke *LCD* sekali lagi agar menjamin data yang dikirimkan ke komputer sama dengan data yang ditampilkan ke *LCD.* Selanjutnya proses pengecekan jumlah proses pencacahan dilakukan untuk mentukan apakah proses akan berlanjut kembali atau tidak.

```
while (1)
      {
      // Place your code here
      if(waktu==0)
        {
          TCCR1B=0x00;
          TCCR2=0x00;
          tot=TCNT2+cacah;
          itoa(tot,tot_lcd);
          for(j=0;j<=strlen(tot_lcd);j++)
              {putchar(tot_lcd[j]);}
          putchar('x');
          lcd_gotoxy(0,1); lcd_puts(tot_lcd);
          cacah=0;
          waktu=rentang;
          if(nomor<jumlah)
              lanjut();
          else
              {
                  lcd_gotoxy(6,3); lcd_puts(" Finish...");
              };
        };
```

Gambar 4.22 *Source code* kondisi ketika waktu pencacahan telah selesai

4.3.9 Proses pencacahan berulang atau *continue*

Dilakukan pengulangan proses pencacahan hingga nomor proses pencacahan yang telah dilakukan = jumlah proses pencacahan yang diminta. Dalam pengulangan proses pencacahan pointer eksekusi program akan berjalan atau *jump* ke fungsi void lanjut. Di dalam void lanjut nomor proses

pencacahan dinaikkan atau *increment*. Selain itu dilakukan pula proses penampilan data ke *LCD*, hal ini dilakukan karena fungsi *lcd_clear()* merupakan pembersihan *LCD* secara keseluruhan. Selanjutnya dilakukan pemberian nilai awal *timer1* dan *counter2* serta aktivasi kembali *timer1* dan *counter2*.

```
189  void lanjut()
190  {
191  nomor++;
192  itoa(nomor,nom);
193  lcd_clear(); lcd_puts("counter :");
194  lcd_gotoxy(0,2); lcd_puts("Time = "); lcd_puts(men);
195  lcd_gotoxy(10,2); lcd_puts("m: "); lcd_puts(det);
196  lcd_gotoxy(15,2); lcd_puts("s");
197  lcd_gotoxy(0,3); lcd_puts("At ");
198  lcd_gotoxy(3,3); lcd_puts(nom);
199  lcd_gotoxy(6,3); lcd_puts("from "); lcd_puts(jum);
200  TCNT2=0x00;
201  TCNT1H=0xE1;
202  TCNT1L=0x54;
203  TCCR2=0x07;
204  TCCR1B=0x05;
205  }
```

Gambar 4.23 *Source code* proses pencacahan selanjutnya

4.3.10 *Finish*

Setelah proses pencacahan telah berakhir dan nomor pencacahan telah sama dengan jumlah proses pencacahan maka tampilan pada jumlah proses pencacahan yang terletak di sudut kanan bawah *LCD* akan muncul tulisan "*Finish...*".

```
325             if(nomor<jumlah)
326                 lanjut();
327             else
328                 {
329                     lcd_gotoxy(6,3); lcd_puts(" Finish...");
330                 };
```

Gambar 4.24 *Source code* kondisi saat sebuah proses pencacahan telah berakhir