DT-AVR

DT-AVR *Application Note*

AN133 – Media Tampilan *7 Segment* Untuk Mikrokontroler AVR®

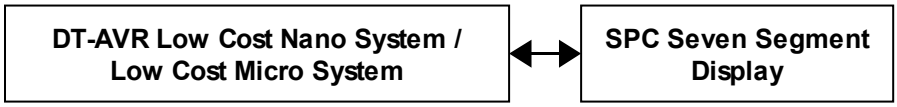
Oleh: Tim IE

Aplikasi ini memberikan contoh penambahan media tampilan *seven segment* pada modul DT-AVR Low Cost Series dengan menggunakan bahasa pemrograman C (CodeVisionAVR©). SPC Seven Segment Display dipilih sebagai media tampilannya agar proses pembuatan *hardware* serta pemrogramannya menjadi lebih mudah dan cepat. AN ini terdiri dari beberapa program yang masing-masing akan memberikan contoh praktis untuk penggunaan tiap antarmuka yang didukung oleh SPC Seven Segment Display. Jalur-jalur komunikasi yang dimiliki oleh SPC Seven Segment Display adalah paralel 4 bit, SPI (*Serial Peripheral Interface*), UART (*Universal Asynchronous Receiver / Transmitter*) RS-232 dan UART RS-485.

Modul-modul yang dibutuhkan dalam aplikasi ini adalah sebagai berikut:

- 1 buah DT-AVR Low Cost Nano System (AT90S2313) / Low Cost Micro System (ATMEGA8535).
- 1 buah SPC Seven Segment Display.
- 1 buah DT-I/O RS232-RS485 Converter.

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



Gambar 1
Blok Diagram AN133

Hubungan antara modul-modul tersebut adalah sebagai berikut:

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Seven Segment Display
GND (J7 / J11 pin 1)	GND (J13 pin 1)
PB0* (J7 / J11 pin 3)	CTRCLR (J13 pin 2)
PB1* (J7 / J11 pin 4)	CTRDIR (J13 pin 3)
PB3* (J7 / J11 pin 6)	CTRCLK (J13 pin 4)

Tabel 1
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display Mode *Stand-alone Counter*

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Seven Segment Display
PD0* (J8 / J13 pin 3)	D0 (J12 pin 3)
PD1* (J8 / J13 pin 4)	D1 (J12 pin 4)
PD2* (J8 / J13 pin 5)	D2 (J12 pin 5)
PD3* (J8 / J13 pin 6)	D3 (J12 pin 6)
PB0* (J7 / J11 pin 3)	CLK/SCK (J12 pin 7)
PB1* (J7 / J11 pin 4)	DOT/MOSI (J12 pin 8)
PB2* (J7 / J11 pin 5)	CLR (J12 pin 9)

Tabel 2
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display secara Paralel 4 Bit

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Seven Segment Display
PB0* (J7 / J11 pin 3)	CLK/SCK (J12 pin 7)
PB1* (J7 / J11 pin 4)	DOT/MOSI (J12 pin 8)
PB2* (J7 / J11 pin 5)	CLR (J12 pin 9)
PB3* (J7 / J11 pin 6)	MISO (J12 pin 10)

Tabel 3
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display secara SPI

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Seven Segment Display
GND (RJ11 pin 3)	SGND (J10 pin 7)
RX (RJ11 pin 5)	TX232RJ (J10 pin 6)
TX (RJ11 pin 4)	RX232RJ (J10 pin 5)

Tabel 4
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display secara UART RS-232

DT-AVR Low Cost Nano System / Low Cost Micro System	DT-I/O RS232 – RS485 Converter
GND (RJ11 pin 3)	COM (J1 pin 3)
RX (RJ11 pin 5)	RXD (J1 pin 5)
TX (RJ11 pin 4)	TXD (J1 pin 4)

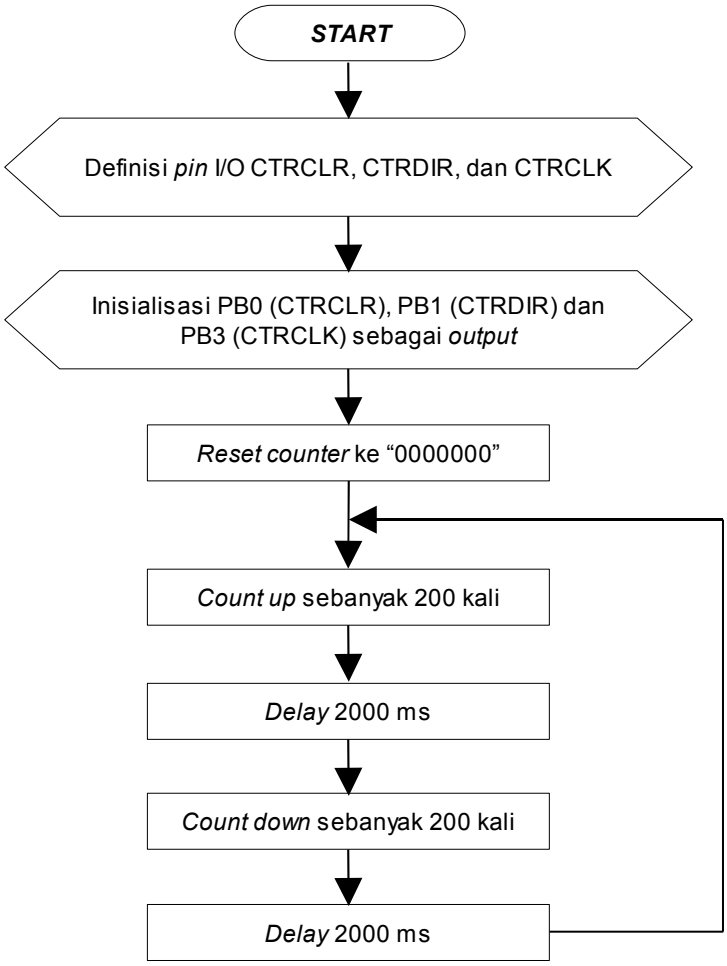
DT-I/O RS232 – RS485 Converter	SPC Seven Segment Display
COM (J18 pin 1)	SGND (J10 pin 4)
D- (J18 pin 2)	B485RJ (J10 pin 1)
D+ (J18 pin 4)	A485RJ (J10 pin 2)

Tabel 5
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display secara UART RS-485 Melalui DT-I/O RS232-RS485 Converter

Pin yang bertanda * pada tabel di atas dapat diganti dengan pin lain tetapi program juga harus disesuaikan. Agar contoh program dapat bekerja dengan baik sesuai dengan jalur komunikasi yang digunakan, maka atur *jumper* S2 (untuk memilih antarmuka atau jenis komunikasi yang digunakan), J11, J8 dan J9 (untuk mengaktifkan mode UART RS-232 / UART RS-485) serta J4, J2 atau J3 (pengaturan *bias* dan *termination* untuk komunikasi UART RS-485) pada SPC Seven Segment Display. Keterangan lebih lengkap terdapat pada Manual SPC Seven Segment Display sub bagian 2.2 (Setting Jumper Mode dan Antarmuka).
Sumber tegangan yang dibutuhkan oleh SPC Seven Segment Display adalah 12 VDC. Hubungkan sumber tegangan 12 VDC ke konektor J1 pada SPC Seven Segment Display. Jika menggunakan sumber tegangan yang berbeda antara DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Seven Segment Display, maka pastikan bahwa jalur *ground* (GND) antara DT-AVR Low Cost Nano System / Low Cost Micro System dan jalur *ground* SPC Seven Segment Display sudah terhubung. Apabila belum terhubung, Anda dapat menghubungkan *ground* DT-AVR Low Cost Series (pin GND, J2) tersebut ke *ground* SPC Seven Segment Display (pin GND, J1). Contoh program untuk komunikasi RS-485 membutuhkan modul DT-I/O RS232-RS485 Converter dengan konfigurasi *jumper* sebagai berikut: J3, J4, J5 posisi 2-3 (mode DTE), J8, J9, J10 posisi 2-3 (mode RS-232), J13 posisi 1-2 (arah otomatis), J12 terpasang dan J11 tidak (*baudrate* 9600 bps). *Jumper* J14, J15, J16 pada DT-I/O RS232-RS485 Converter diatur untuk *bias* dan *termination* pada jalur RS-485.

Setelah semua rangkaian dan sumber tegangan terhubung dengan benar, programlah “KCOUNTER.C” atau “K4BIT.C” atau “KSPI.C” atau “KRS232.C” atau “KRS485.C” (dengan melakukan *compile* atau *make* program tersebut terlebih dahulu pada *project* “SPC 7Segment.prj”) ke DT-AVR Low Cost Nano System / Low Cost Micro System menggunakan DT-HiQ AVR In System Programmer atau *in-system programmer* lainnya. Apabila menggunakan antarmuka SPI dengan contoh program ”KSPI.C” dan UART RS-485 dengan contoh program ”KRS485.C”, maka SPC Seven Segment Display harus diisi dengan alamat 53 terlebih dahulu.

Flowchart dari program “KCOUNTER.C” adalah sebagai berikut:



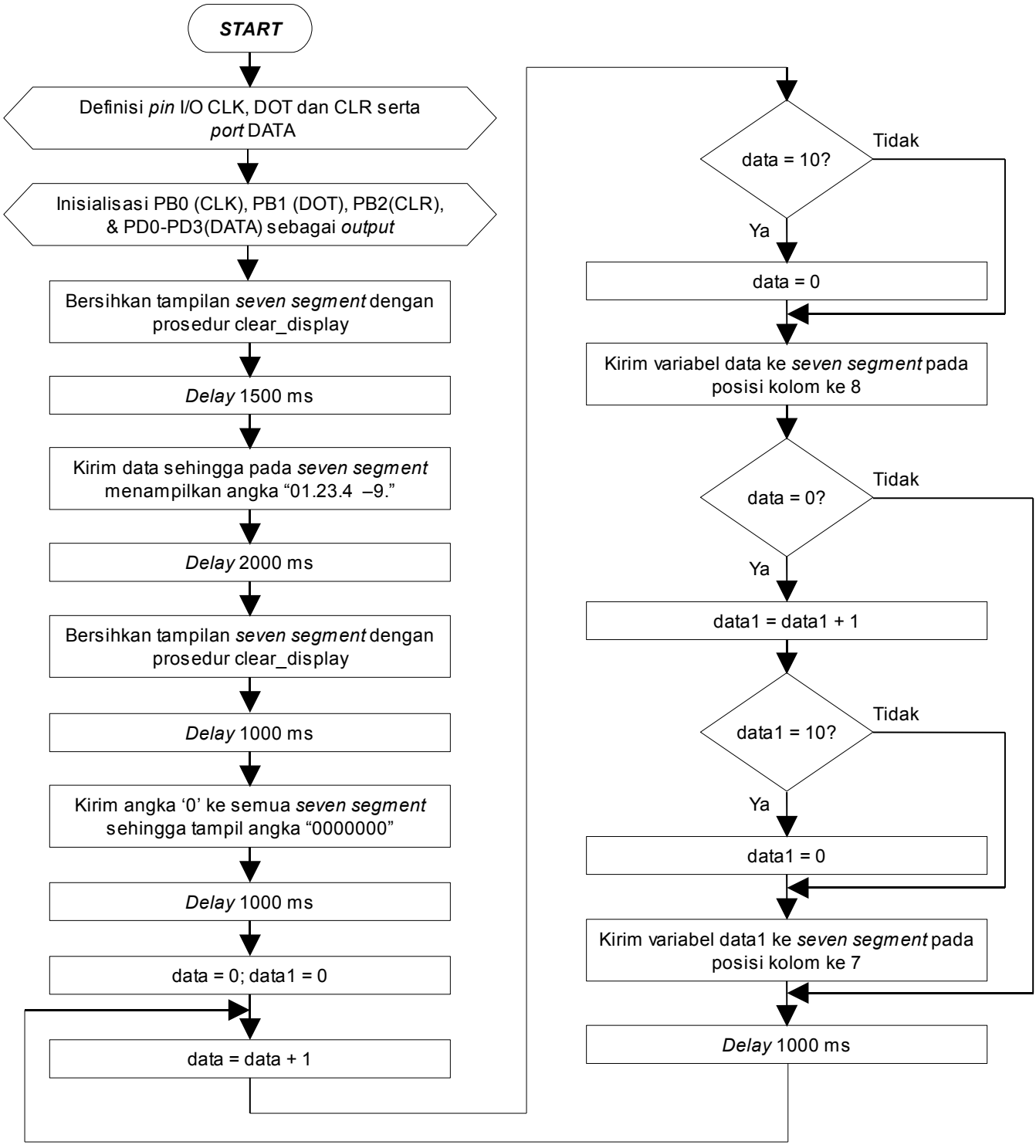
Gambar 2
***Flowchart* Program untuk “KCOUNTER.C”**

Program utama (KCOUNTER.C) akan diproses sebagai berikut:

1. Proses yang pertama kali dilakukan adalah menentukan definisi pin I/O CTRCLR, CTRDIR dan CTRCLK untuk komunikasi dengan SPC Seven Segment Display dalam mode *Stand-alone Counter*.
2. Proses berikutnya yaitu inisialisasi pin I/O PB0 (CTRCLR), PB1 (CTRDIR) dan PB3 (CTRCLK) sebagai jalur *output*.
3. Setelah proses inisialisasi selesai, berikutnya program menolkan (*reset*) *counter* SPC Seven Segment Display sehingga tampilan pada *seven segment* menjadi “00000000”. Proses *reset counter* ini dilakukan dengan cara *pin* CTRCLR diberi logika '0', setelah 30 mikro detik *pin* CTRCLK diberi logika '0' dan setelah 200 mikro detik *pin* CTRCLK dan CTRCLR diberi logika '1'.
4. Setelah itu program akan memberi logika ‘0’ pada pin CTRDIR sehingga SPC Seven Segment Display akan menghitung naik (*Count Up*). Kemudian program akan memberikan pulsa dengan lebar ±100 mili detik sebanyak 200 kali ke *pin* CTRCLK sehingga tampilan pada *seven segment* pada akhirnya akan menjadi “00000200”. Lalu *delay* selama 2000 mili detik.
5. Proses berikutnya sama dengan langkah pada nomor 4 tetapi dalam hitungan turun (*Count Down*, *pin* CTRDIR = ‘1’) sehingga tampilan pada *seven segment* pada akhirnya akan menjadi “00000000”. Langkah nomor 4 dan 5 diulang secara terus-menerus.

Listing program “KCOUNTER.C” terdapat pada **AN133.ZIP**.

Flowchart dari program "K4BIT.C" adalah sebagai berikut:



Gambar 3
***Flowchart* Program untuk "K4BIT.C"**

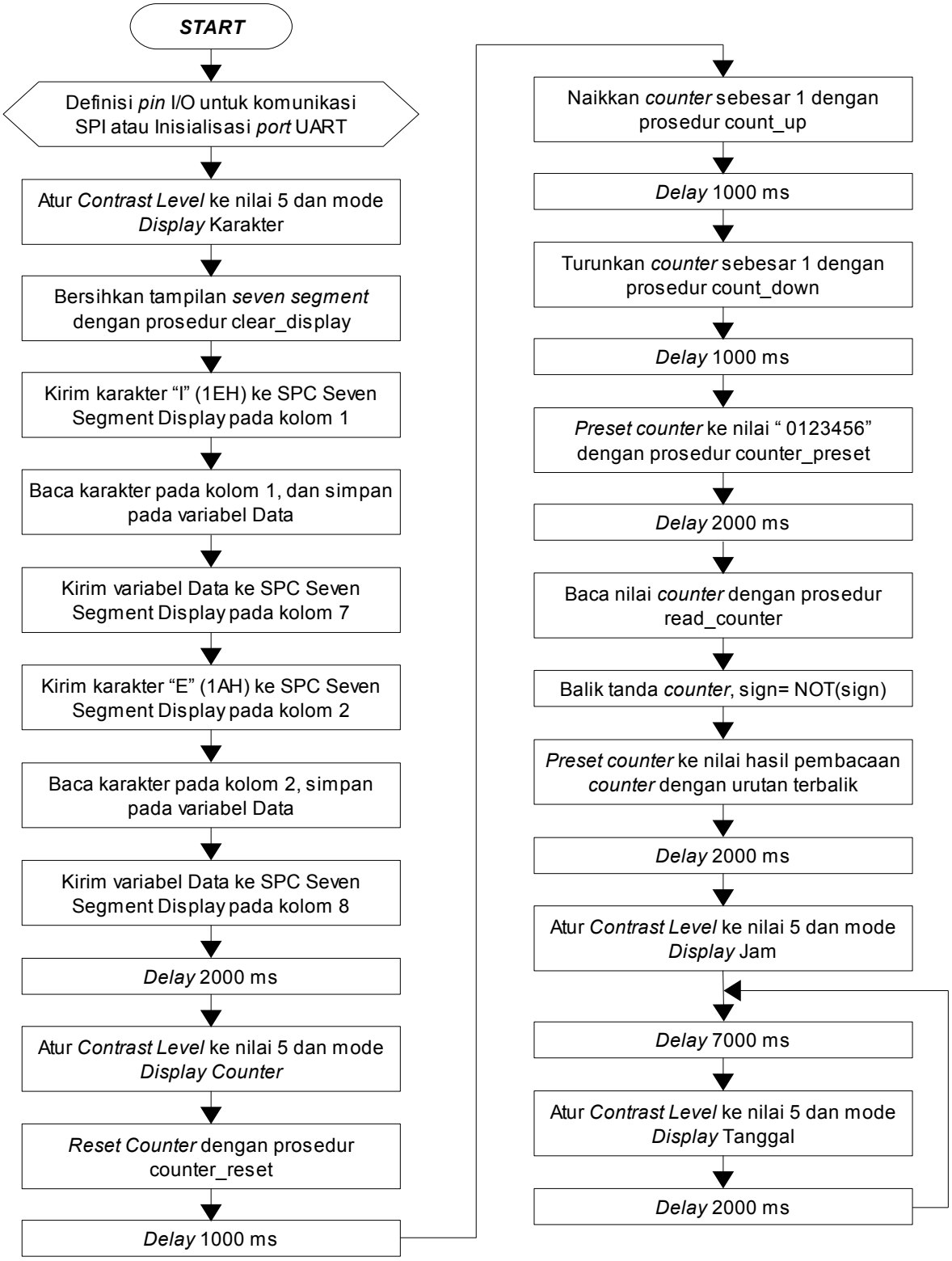
Program utama (K4BIT.C) akan diproses sebagai berikut:

1. Proses yang pertama kali dilakukan adalah menentukan definisi *pin* I/O CLK, DOT dan CLR serta *port* DATA untuk berkomunikasi secara paralel 4 bit dengan SPC Seven Segment Display.
2. Proses berikutnya adalah inisialisasi PB0 (terhubung ker *pin* CLK), PB1 (terhubung ke *pin* DOT), PB2 (terhubung ke *pin* CLR) dan PD0-PD3 (terhubung ke *port* DATA, D0-D3) sebagai *output*.

3. Setelah proses inisialisasi selesai, berikutnya program akan membersihkan tampilan *seven segment* dengan cara mengirimkan karakter kosong 0Bh ke semua kolom *seven segment* sehingga tidak tampil apa-apa pada tampilan *seven segment*. Proses ini terdapat pada prosedur clear_display. Kemudian *delay* selama 1500 mili detik.
4. Setelah itu program akan mengirimkan karakter ke SPC Seven Segment Display sehingga pada tampilan *seven segment* tampil “01.23.4 –9”. Pengiriman karakter ini dilakukan satu-persatu mulai dari kolom pertama sampai kolom kedelapan. Titik (*dot*) akan menyala pada kolom karakter dengan kondisi *pin* DOT (variabel dot_cond) berlogika ‘1’, kondisi variabel dot_cond ini diatur ketika mengirimkan karakter menggunakan prosedur send_4_bit_data. Setelah pengiriman semua karakter selesai, program berhenti selama 2000 mili detik.
5. Proses berikutnya sama dengan langkah nomor 3 yaitu *clear display* tetapi diikuti dengan *delay* selama 1000 mili detik.
6. Setelah langkah nomor 5 selesai, program akan mengirimkan karakter “0” ke semua kolom tampilan *seven segment* sehingga tampil “0000000”. Lalu *delay* selama 1000 mili detik.
7. Proses yang terakhir adalah *counter up* (penghitung naik) selebar 2 digit (kolom 7 dan 8) dengan lama kenaikan *counter* tiap 1 detik. *Counter* dimulai dari “00000000”, “00000001”, “00000002”, …, “00000009”, “00000010”, “00000011”, … sampai “00000099”. Setelah *counter* mencapai “00000099” maka *counter* akan kembali lagi ke “00000000”, lalu “00000001”, dan seterusnya. Program *counter* 2 digit ini dilakukan berulang-ulang.

L*isting* program “K4BIT.C” terdapat pada **AN133.ZIP**.

Flowchart dari program “KSPI.C”, “KRS232.C” dan “KRS485.C” adalah sama (nama fungsi / prosedur pada contoh program adalah sama, hanya berbeda pada cara pengiriman karakter / perintah), yaitu sebagai berikut:



Gambar 4
***Flowchart* Program Untuk “KSPI.C”, “KRS232.C” dan “KRS485.C”**

Program utama (KSPI.C, KRS232.C dan KRS485.C) akan diproses sebagai berikut:

1. Proses yang pertama kali dilakukan adalah menentukan definisi *pin* I/O dan inisialisasi *pin* I/O. Program untuk antarmuka SPI menggunakan *pin* I/O PB0 (SCK), PB1 (MOSI), PB2 (CLR) dan PB3 (MISO) dengan inisialisasi awal semua *pin* sebagai *output* dan berlogika '1', kecuali *pin* PB3 sebagai *input*. Program untuk antarmuka UART (RS-232 atau RS-485) melakukan inisialisasi *port* UART pada *baud rate* 9600 bps, 8 bit data, tanpa bit *parity*, 1 bit stop dan tanpa *flow control.*
2. Setelah proses inisialisasi selesai, berikutnya program akan mengirimkan nilai *contrast level* sebesar 5 dan mode *display* karakter. Nilai *contrast level* dan mode dikirim menggunakan prosedur set_display. Prosedur set_display juga digunakan untuk menjalankan RTC (variabel RTC='1') dan menentukan kondisi *display* tidak berkedip (variabel blink = '0').
3. Kemudian program mengirimkan perintah *clear display* untuk membersihkan tampilan *seven segment* dengan menggunakan prosedur clear_display.
4. Setelah itu program akan mengirimkan karakter “I” (1EH) ke tampilan *seven segment* pada kolom 1 dengan menggunakan prosedur write_character. Kemudian membacanya kembali dengan menggunakan prosedur read_character. Hasil pembacaan dikirimkan kembali ke tampilan *seven segment* pada kolom 7.
5. Seperti pada langkah nomor 3, program akan mengirimkan karakter “E” (1AH) ke tampilan *seven segment* pada kolom 2, kemudian membacanya kembali dan hasil pembacaan dikirimkan kembali ke tampilan *seven segment* pada kolom 8. Setelah itu *delay* selama 2 detik.
6. Setelah program pada langkah nomor 5 selesai, program akan mengirimkan nilai *contrast level* sebesar 5 dan mode *display counter* dengan menggunakan prosedur set_display.
7. Selanjutnya program mengirimkan perintah *counter reset* dengan menggunakan prosedur counter_reset. Sehingga tampilan *seven segment* menjadi “0000000”. Kemudian delay selama 1 detik.
8. Program mengirimkan perintah *count up* dengan menggunakan prosedur count_up. Setelah itu *delay* selama 1 detik. Tampilan *seven segment* akan menampilkan “0000001”.
9. Program mengirimkan perintah *count down* dengan menggunakan prosedur count_down. Setelah itu *delay* selama 1 detik. Tampilan *seven segment* akan kembali menampilkan “0000000”.
10. Selanjutnya program melakukan *preset counter* ke nilai “0123456” menggunakan prosedur counter_preset. Tampilan pada *seven segment* menjadi “ 0123456”, setelah itu *delay* selama 2 detik.
11. Program akan membaca nilai *counter* dari SPC Seven Segment Display (yaitu bernilai “0123456”) juga tanda (variabel Sign) *counter* yaitu bertanda positif (berlogika ‘0’). Proses ini terdapat pada prosedur read_counter. Selanjutnya tanda *counter* dibalik menjadi negatif.
12. Setelah membalik tanda *counter* (variabel Sign berlogika ‘1’), program akan mengirimkan kembali nilai *counter* yang telah dibaca tetapi dengan urutan digit yang terbalik. Tampilan pada *seven segment* menjadi “-6543210”. Setelah itu *delay* selama 2 detik.
13. Proses yang terakhir adalah menampilkan mode *display* jam selama 7 detik kemudian berganti ke mode *display* tanggal selama 2 detik. Langkah ini dilakukan secara berulang-ulang.

Listing program “KSPI.C”, “KRS232.C” dan “KRS485.C” terdapat pada **AN133.ZIP**.

Pada **AN133.ZIP** juga disertakan contoh program untuk digunakan pada DT-AVR Low Cost Micro System (pada direktori “Program ATMEGA8535”). Pada prinsipnya *flowchart* dan isi program sama dengan contoh program di atas hanya berbeda pada penggunaan *file* definisi untuk mikrokontroler (mega8535.h) dan berbeda pada inisialisasi UART (untuk mikrokontroler ATMEGA8535 disebut sebagai USART), terutama untuk contoh program yang menggunakan komunikasi UART RS-232 dan UART RS-485.

Selamat Berinovasi!