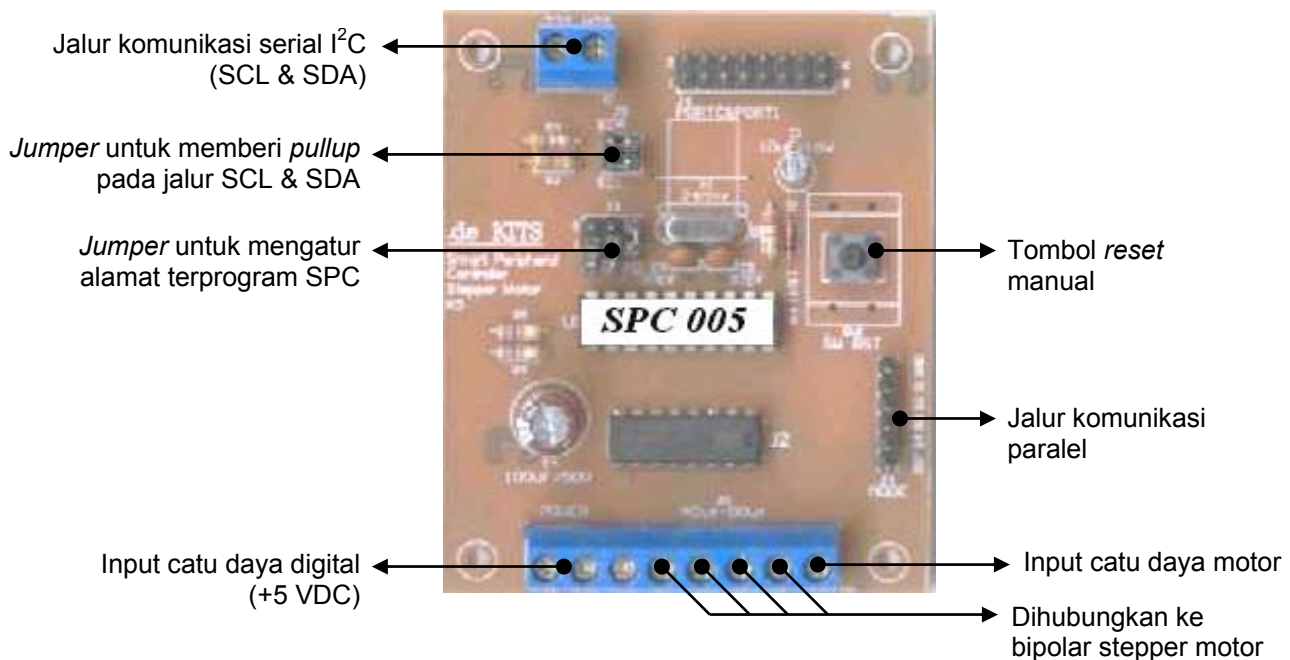


Motor stepper seringkali kita gunakan untuk aplikasi robotika, karena poros motor stepper dapat digerakkan dengan sudut putar tertentu tanpa harus menggunakan umpan balik posisi. Berikut ini adalah aplikasi sederhana untuk motor stepper bipolar yang dikendalikan melalui SPC Stepper Motor. Modul kontroler dalam aplikasi ini menggunakan DT-AVR Low Cost Nano System / Low Cost Micro System dan bahasa pemrograman BASIC dengan *compiler* BASCOM-AVR[®]. Aplikasi ini akan memberi contoh komunikasi SPC Stepper Motor dengan DT-AVR Low Cost Series baik secara serial I²C maupun paralel. Aplikasi ini dapat dikembangkan menjadi suatu sistem pengendali motor stepper bipolar dalam sebuah robot atau peralatan otomatis.

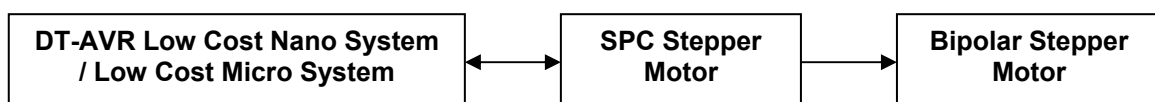
Komponen yang diperlukan:

- 1 DT-AVR Low Cost Nano System / Low Cost Micro System
- 1 SPC Stepper Motor
- 1 motor stepper bipolar



Gambar 1
SPC Stepper Motor

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



Gambar 2
Blok Diagram AN80

Hubungan antara modul-modul tersebut secara I²C adalah sebagai berikut:

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Stepper Motor
+5VDC	+5V
GND	GND
PB.4*	SCL (J1)
PB.5*	SDA (J1)

* Pin ini tidak mutlak dan dapat diganti pin lain tetapi harus mengubah program

Tabel 1
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Stepper Motor secara Serial I²C

Pada SPC Stepper Motor, pasanglah *jumper* J2 untuk memberi resistor *pull up* pada jalur SCL & SDA dan atur *jumper* J3 untuk alamat terprogram 000b. Kemudian hubungkan bipolar stepper motor pada SPC Stepper Motor sesuai petunjuk dalam manual SPC Stepper Motor.

Setelah semua modul dan sumber tegangan terhubung dengan benar, programlah `I2csteppermotor.bin` atau `I2csteppermotor.hex` ke dalam DT-AVR Low Cost Nano System / Low Cost Micro System menggunakan DT-HiQ AVR In System Programmer atau divais ISP programmer lain dengan konektor *header* 10-pin standar Atmel.

Hubungan antara modul-modul tersebut secara Paralel adalah sebagai berikut:

DT-AVR Low Cost Nano System / Low Cost Micro System	SPC Stepper Motor
+5VDC	+5V
GND	GND
PB.0*	Sel (S1-J4)
PB.1*	Dir (S2-J4)
PB.2*	Mode (S3-J4)
PB.3*	Step (S4-J4)
PB.4*	Reset (RST-J4)

* Pin ini tidak mutlak dan dapat diganti pin lain tetapi harus mengubah program

Tabel 2
Hubungan DT-AVR Low Cost Nano System / Low Cost Micro System dengan SPC Stepper Motor secara Paralel

Untuk komunikasi secara paralel, hubungan antara DT-AVR Low Cost Nano System / Low Cost Micro System dan SPC Stepper Motor mengikuti Tabel 2. Setelah semua rangkaian dan sumber tegangan terhubung dengan benar, programlah `parallelstepper.bin` atau `parallelstepper.hex` ke dalam DT-AVR Low Cost Nano System / Low Cost Micro System menggunakan DT-HiQ AVR In System Programmer atau divais ISP programmer lain dengan konektor *header* 10-pin standar Atmel.

Catatan:

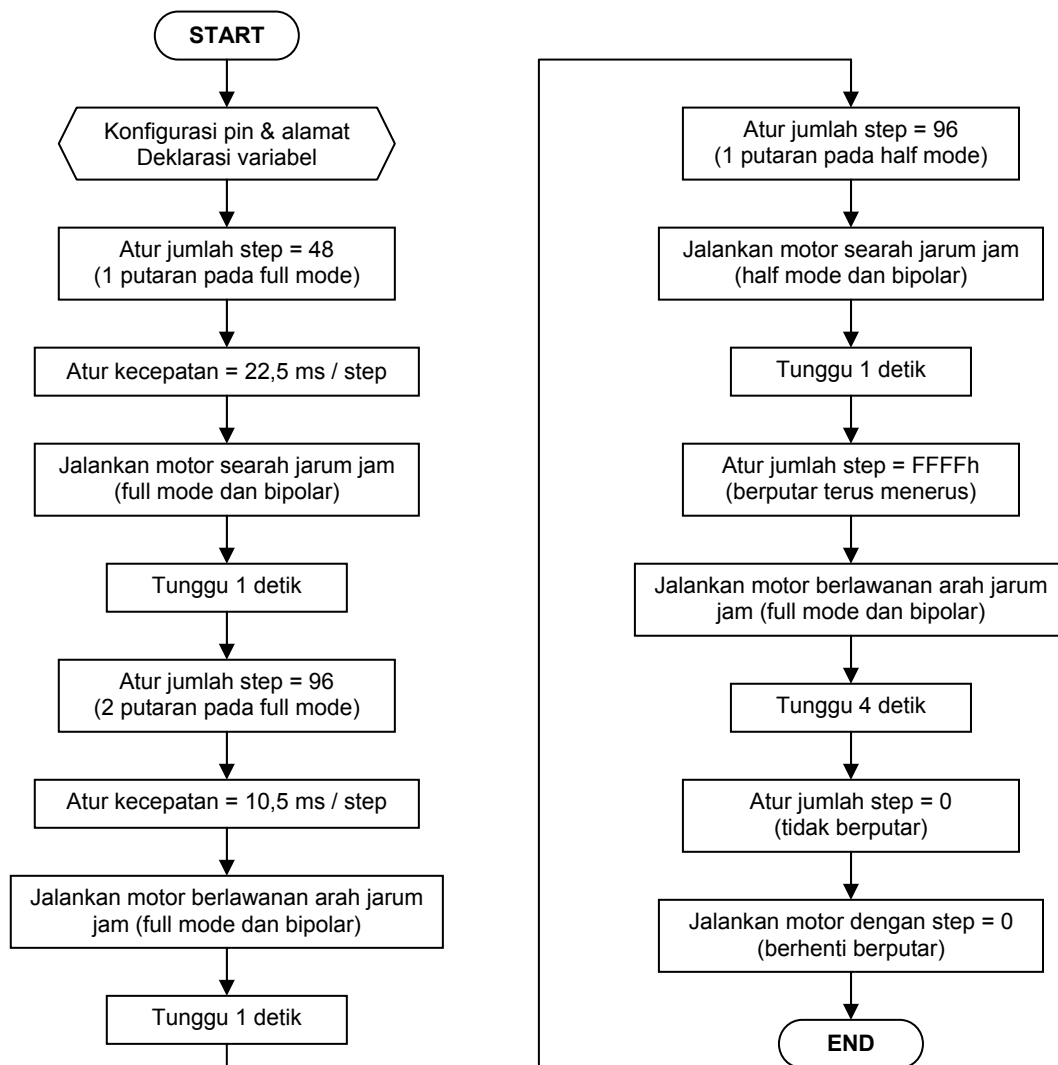
Dalam aplikasi ini menggunakan DT-AVR Low Cost Nano System (dengan mikrokontroler AT90S2313) sehingga `I2csteppermotor.bin` / `I2csteppermotor.hex` maupun `parallelstepper.bin` / `parallelstepper.hex` yang disertakan dalam **AN80.ZIP** hanya berlaku untuk AT90S2313.

Apabila menggunakan mikrokontroler tipe lain, bukalah `I2csteppermotor.bas` atau `parallelstepper.bas` menggunakan BASCOM-AVR[®] dan ubahlah baris pertama dalam program tersebut:

- \$regfile = "2313def.dat" '→ untuk AT90S2313
- \$regfile = "8535def.dat" '→ untuk AT90S8535
- \$regfile = "m8535.dat" '→ untuk ATmega8535 (DT-AVR Low Cost Micro System)

agar sesuai dengan tipe mikrokontroler yang akan digunakan. Lalu *compile* ulang program agar menghasilkan *file* dengan ekstensi `.bin` atau `.hex` yang sesuai dan dapat diprogram ke dalam modul mikrokontroler.

Flowchart program untuk komunikasi secara serial I²C adalah sebagai berikut:



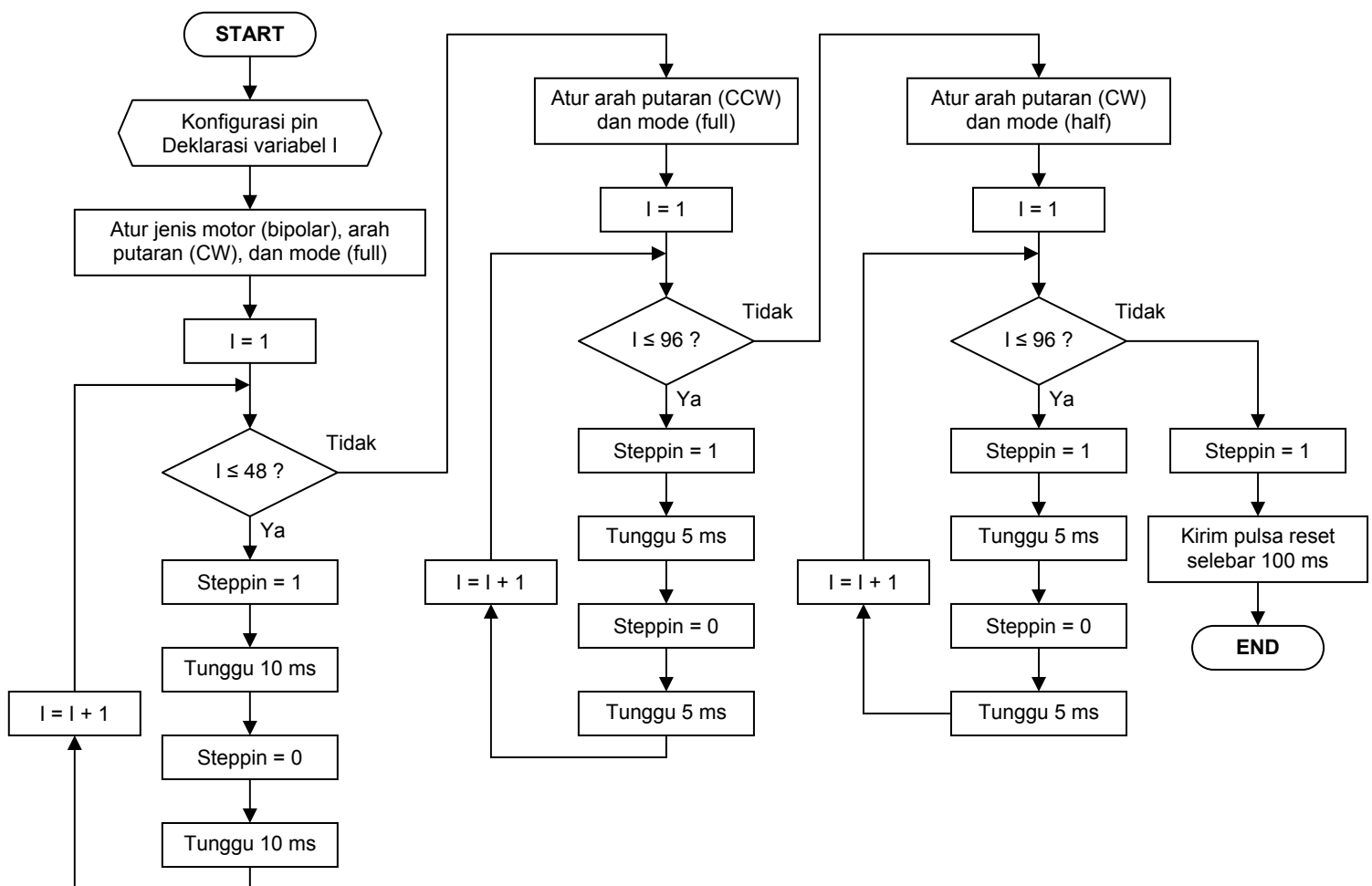
Gambar 3
FlowChart Program untuk Komunikasi secara Serial I²C

Program utama untuk komunikasi secara serial I²C akan diproses sebagai berikut:

1. Proses yang pertama dilakukan adalah menentukan definisi pin I/O (SDA dan SCL) untuk komunikasi I²C serta penetapan alamat SPC Stepper Motor yaitu E0h.
2. Kemudian deklarasi variabel Step1, Step2, Speed1, Speed2, Command.
Step1 adalah variabel Step 8 bit MSB.
Step2 adalah variabel Step 8 bit LSB.
Speed1 adalah variabel Speed 8 bit MSB.
Speed2 adalah variabel Speed 8 bit LSB.
Command adalah variabel yang berisi konfigurasi motor stepper yang dipakai dalam prosedur Cmd_Stepper.
I, J, K adalah variabel yang digunakan dalam proses pengulangan.
3. Step1 diisi dengan 00h dan Step2 diisi dengan 48 desimal yang kemudian diinisialisasikan ke dalam SPC Stepper Motor, agar apabila motor diaktifkan maka akan bergerak sebanyak 48 step. (48 step = 1 putaran apabila motor dalam keadaan *full mode*)
4. Speed1 diisi dengan 50h dan Speed2 diisi dengan 00h yang kemudian diinisialisasikan ke dalam SPC Stepper Motor, agar apabila motor diaktifkan maka akan bergerak dengan kecepatan 22,5 ms/step (lihat manual SPC Stepper Motor).
5. Motor diaktifkan dengan konfigurasi bipolar, searah jarum jam, dan *full mode*.
6. Tunggu selama 1 detik.

7. Step1 diisi dengan 00h, Step2 diisi dengan 96 desimal dan diinisialisasikan agar apabila motor diaktifkan maka akan bergerak sebanyak 96 step. (96 step = 2 putaran dalam keadaan full mode)
8. Speed1 diisi dengan B0h, Speed2 diisi dengan 00h dan diinisialisasi sehingga apabila motor diaktifkan maka akan bergerak dengan kecepatan 10.5 ms/step.
9. Motor diaktifkan dengan konfigurasi bipolar, berlawanan arah jarum jam dan full mode.
10. Tunggu selama 1 detik.
11. Step1 (00h) dan Step2 (96 desimal) diinisialisasikan ke dalam SPC Stepper Motor lagi.
12. Motor diaktifkan lagi dengan konfigurasi bipolar, searah jarum jam dan half mode (96 step menjadi hanya 1 putaran namun dengan torsi yang lebih besar).
13. Tunggu 1 detik.
14. Kemudian Step1 diisi dengan FFh dan Step2 diisi juga dengan FFh agar apabila motor diaktifkan maka akan berputar terus menerus hingga mendapatkan perintah stop (Step1 dan Step2 diberi nilai nol).
15. Setelah itu, motor diaktifkan dengan konfigurasi bipolar, searah jarum jam, dan full mode.
16. Tunggu 4 detik.
17. Step1 diisi dengan 00h, Step2 diisi dengan 00h, dan diinisialisasikan ke dalam SPC.
18. Motor diaktifkan dengan konfigurasi bipolar, searah jarum jam, dan full mode. Perintah ini akan mengakibatkan motor stepper berhenti bergerak karena Step1 dan Step2 bernilai nol.

Flowchart program untuk komunikasi secara paralel adalah sebagai berikut:



Gambar 4
Flowchart Program untuk Komunikasi secara Paralel

Program untuk komunikasi secara paralel akan diproses sebagai berikut:

1. Konfigurasi pin yang akan digunakan untuk komunikasi dengan SPC Stepper Motor secara paralel dan pin Rst (PB.4) diberi logika 'low' untuk mengaktifkan SPC Stepper Motor. Serta deklarasi variabel I yang akan dipakai dalam proses looping.

2. Penetapan jenis motor stepper (bipolar), arah putar (searah jarum jam), mode putaran (*full mode*).
3. Setelah itu, program akan mengeluarkan pulsa pada pin Steppin (PB.3) dengan periode 20 ms sebanyak 48 kali sehingga motor bergerak sebanyak satu putaran (1 putaran = 48 step dalam *full mode*).
4. Setelah motor selesai bergerak, program menetapkan lagi arah putar motor yaitu berlawanan arah jarum jam dengan mode putaran tetap (*full mode*).
5. Program akan mengeluarkan pulsa pada pin Steppin dengan periode 10 ms sebanyak 96 kali sehingga motor bergerak sebanyak dua putaran.
6. Setelah motor selesai bergerak, program menetapkan lagi arah putar motor yaitu searah jarum jam dengan mode putaran *half mode*.
7. Program akan mengeluarkan pulsa pada pin Steppin dengan periode 10 ms sebanyak 96 kali sehingga motor bergerak sebanyak satu putaran (karena dalam *half mode*). Lalu pin step diberi logika 'high'.
8. Setelah berhenti, motor stepper akan berada dalam keadaan terkunci. Dan program akan melakukan *reset* pada SPC Stepper Motor dengan memberi pulsa 'high' selebar 100 ms pada pin Rst untuk melepas motor stepper dari keadaan terkunci.

Listing program terdapat pada **AN80.ZIP**.

Selamat berinovasi!

BASCOM-AVR is copyright by MCS Electronics.
I2C is a Registered Trademark of Philips Semiconductors