SPC

SMART PERIPHERAL CONTROLLER

MOTOR CONTROLLER

Daftar Isi

1. PENDAHULUAN

Smart Peripheral Controller / SPC MOTOR CONTROLLER merupakan sebuah modul pengendali motor DC dan motor stepper yang mampu digunakan untuk mengendalikan kecepatan dan arah putaran 4 buah motor DC atau 2 buah motor stepper. Modul ini dilengkapi dengan pengendali PID (*Proportional Integral Differential*) untuk kendali motor DC yang bisa diatur (*tuning*) sendiri oleh pengguna. Selain itu modul SPC MOTOR CONTROLLER ini juga dilengkapi dengan kemampuan *microstepping* untuk motor stepper sehingga gerakan motor stepper dapat lebih mulus dan tidak patah-patah tanpa mengurangi kemampuan torsi motor stepper.

1.1. SPESIFIKASI SPC MOTOR CONTROLLER

Spesifikasi SPC MOTOR CONTROLLER sebagai berikut:

- Daya diperoleh dari sumber catu daya dengan tegangan 9 – 12 Volt.
- Mampu digunakan untuk mengendalikan kecepatan dan arah putaran 4 motor DC atau 2 buah motor stepper.
- Umpan balik *speed encoder* (frekuensi maksimum 10 kHz).
- Mampu mendeteksi perubahan kecepatan dengan ketelitian hingga 0,1 RPS (*rotation per second*) atau 6 RPM (*rotation per minute*).
- Kendali motor DC dapat dilakukan secara *close loop* PID maupun *open loop*.
- Parameter PID, toleransi kesalahan, dan waktu *sampling* dapat diatur.
- Mendukung tipe *step* (untuk *motor stepper*): Fullstep, Halfstep, Microstep4, Microstep8, Microstep16, dan Microstep32.
- Kecepatan motor stepper sampai dengan 255 PPS (*pulse per second*).
- Gerak motor stepper dengan ketelitian sampai dengan 0,1° (tergantung tipe *step* dan ketelitian motor stepper).
- Tersedia antarmuka UART RS232/TTL, I^2C, dan Lebar Pulsa.
- Pin Input/Output kompatibel dengan level tegangan TTL dan CMOS.
- Kompatibel dengan modul-modul EMS H-Bridge.
- Terdapat 4 set output PWM (*Pulse Width Modulation*) 8-bit dengan frekuensi 600 Hz.
- Terdapat 4 kanal ADC 8-bit dengan tegangan referensi 5 Volt dan *sampling rate* maksimum 25 kHz.
- Terdapat 4 pin *General Purpose Input/Output* (GP I/O).

1.2. SISTEM YANG DIANJURKAN

Sistem yang dianjurkan untuk penggunaan SPC MOTOR CONTROLLER adalah:

<u>Perangkat keras:</u>

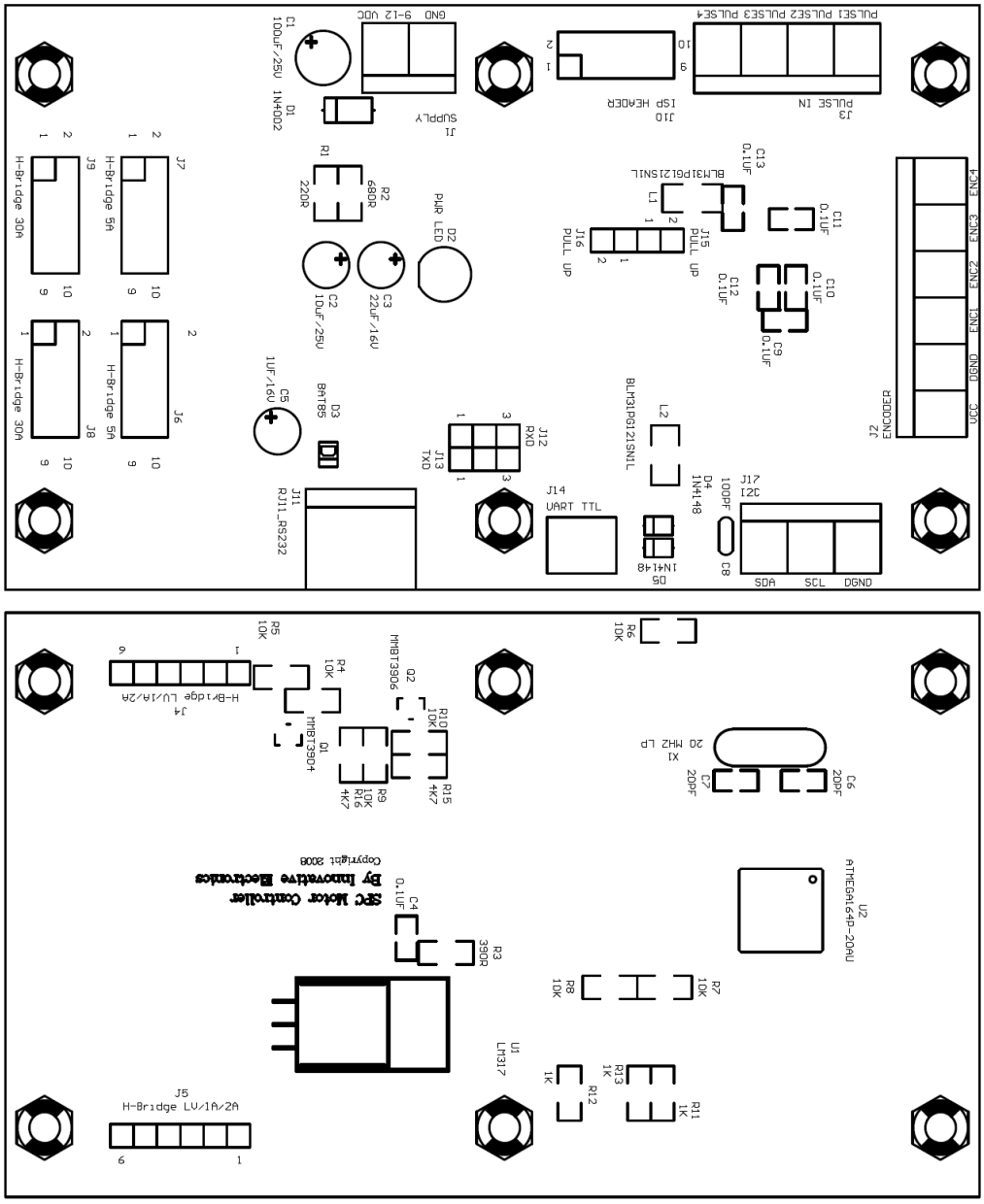
- PC™ AT™ Pentium® IBM™ Compatible dengan *port* Serial (COM1/COM2) dan Paralel (LPT).
- DT-51 Minimum System, DT-51 Low Cost Series, atau DT-AVR Low Cost Series.
- CD-ROM Drive dan Hard disk.

<u>Perangkat lunak:</u>

- Sistem operasi Windows® 98 SE.
- BASCOM-8051©, BASCOM-AVR©, atau CodeVisionAVR©.
- *File* yang ada pada pada CD program:
  DUASTEPPER.PRJ, DUASTEPPER.C, DUASTEPPER.HEX, MANUAL SPC MOTOR CONTROLLER dan QUICK START SPC MOTOR CONTROLLER.

2. PERANGKAT KERAS SPC MOTOR CONTROLLER

2.1. TATA LETAK KOMPONEN SPC MOTOR CONTROLLER



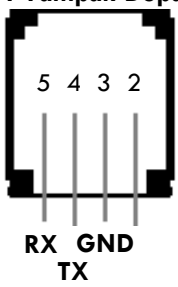
2.2. KONEKTOR DAN PENGATURAN JUMPER

Konektor J1 berfungsi sebagai konektor untuk catu daya modul.

Pin	Nama	Fungsi
1	GND	Titik referensi untuk catu daya input
2	VIN	Terhubung ke catu daya untuk input (9 – 12 Volt)

Konektor RJ11 RS232 (J11) berfungsi sebagai konektor untuk antarmuka UART RS-232.

J11 Tampak Depan

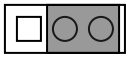


Pin	Nama	Fungsi
2	NC	Tidak terhubung kemana-mana
3	GND	Titik referensi Ground
4	TX	Input serial level RS-232 ke modul
5	RX	Output serial level RS-232 dari modul

Konektor UART TTL (J14) berfungsi sebagai konektor untuk antarmuka UART TTL.

Pin	Nama	Fungsi
1	RXTTL	Input serial level TTL ke modul
2	TXTTL	Output serial level TTL dari modul
3	GND	Titik referensi Ground

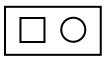
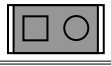
Jumper RXD dan TXD (J12 dan J13) berfungsi untuk memilih level tegangan antarmuka UART yang digunakan oleh SPC MOTOR CONTROLLER.

***Jumper* RXD dan TXD J12 dan J13**	**Level Tegangan UART**
1 2 3 	UART RS-232
1 2 3 	UART TTL

Konektor I2C (J17) berfungsi sebagai konektor untuk antarmuka I^2C.

Pin	Nama	Fungsi
1	SDA	I^2C-bus data – input / output
2	SCL	I^2C-bus *clock* – input
3	GND	Titik referensi Ground

Jumper PULL-UP (J15 dan J16) berfungsi untuk mengaktifkan resistor *pull-up* untuk pin SDA dan SCL pada antarmuka I^2C. Berikut deskripsi bagaimana mengaktifkan resistor *pull-up* untuk pin SDA dan SCL melalui J15 dan J16:

***Jumper* PULL-UP J15 dan J16**	**Fungsi**
1 2 	*Pull-up* tidak aktif (*jumper* terlepas)
1 2 	*Pull-up* aktif (*jumper* terpasang)

Penting !
Apabila lebih dari satu modul dihubungkan pada I^2C-bus maka *jumper* J15 dan J16 (SCL/SDA) salah satu modul saja yang perlu dipasang.

Konektor ENCODER (J2) berfungsi sebagai konektor untuk input umpan balik dari *speed encoder* yang akan digunakan untuk mengukur kecepatan masing-masing motor DC (lihat **bagian 2.3**).

Pin	Nama	Fungsi
1	VCC	Output tegangan +5 Volt
2	GND	Titik referensi Ground
3	ENC1	Input level TTL dari *speed encoder* yang terhubung dengan motor DC ke-1
4	ENC2	Input level TTL dari *speed encoder* yang terhubung dengan motor DC ke-2
5	ENC3	Input level TTL dari *speed encoder* yang terhubung dengan motor DC ke-3
6	ENC4	Input level TTL dari *speed encoder* yang terhubung dengan motor DC ke-4

Konektor PULSE (J3) berfungsi sebagai konektor untuk input lebar pulsa yang dapat digunakan untuk menentukan arah dan kecepatan putaran motor DC (lihat **bagian 4**).

Pin	Nama	Fungsi
1	PULSE IN1	Input lebar pulsa untuk motor DC ke-1
2	PULSE IN2	Input lebar pulsa untuk motor DC ke-2
3	PULSE IN3	Input lebar pulsa untuk motor DC ke-3
4	PULSE IN4	Input lebar pulsa untuk motor DC ke-4

Konektor H-BRIDGE LV/1A/2A (J4 dan J5) merupakan **satu** set output untuk dihubungkan ke rangkaian *driver* motor DC atau motor stepper. **Konektor ini memiliki susunan dan letak yang kompatibel dengan modul EMS LOW VOLTAGE DUAL H-BRIDGE, EMS 1 A DUAL H-BRIDGE, dan EMS 2 A DUAL H-BRIDGE**.

J4			
Pin	**Nama**	**I/O**	**Fungsi**
1	M1DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-1** atau lilitan **stepper ke-1** (bersama-sama dengan M1DIR2) menggunakan modul EMS H-Bridge
2	M1DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-1** atau lilitan **stepper ke-1** (bersama-sama dengan M1DIR1) menggunakan modul EMS H-Bridge
3	M1CS	I	Input tegangan analog ADC ke-1
4	M1PWM	O	Output PWM ke motor DC 1
5	VCC	-	Tegangan output ke modul EMS H-Bridge (5 Volt)
6	GND	-	Titik referensi Ground

J5			
Pin	**Nama**	**I/O**	**Fungsi**
1	M2DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-2** atau lilitan **stepper ke-1** (bersama-sama dengan M2DIR2) menggunakan modul EMS H-Bridge
2	M2DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-2** atau lilitan **stepper ke-1** (bersama-sama dengan M2DIR1) menggunakan modul EMS H-Bridge
3	M2CS	I	Input tegangan analog ADC ke-2
4	M2PWM	O	Output PWM ke motor DC 2
5	VCC	-	Tegangan output ke modul EMS H-Bridge (5 Volt)
6	GND	-	Titik referensi Ground

Berikut ini contoh hubungan antara SPC MOTOR CONTROLLER dengan EMS LOW VOLTAGE DUAL H-BRIDGE, EMS 1 A DUAL H-BRIDGE, dan EMS 2 A DUAL H-BRIDGE:

Pin	**SPC MOTOR CONTROLLER J4**	**EMS Low Voltage Dual H-Bridge J1**	**EMS 1 A Dual H-Bridge J1**	**EMS 2 A Dual H-Bridge J3**	**Catatan**
1	M1DIR1	M1D1	M1IN1	M1IN1	
2	M1DIR2	M1D2	M1IN2	M1IN2	
3	M1CS	NC	NC	M1CS	
4	M1PWM	M1PWM	M1EN	M1EN	
5	VCC	VCC	VCC	VCC	Karena pin ini terhubung, maka catu daya digital (+5V) untuk modul EMS H-Bridge **tidak** boleh diperoleh dari sumber lain
6	GND	PGND	PGND	PGND	Pin ini harus dihubungkan

Pin	SPC MOTOR CONTROLLER J5	EMS Low Voltage Dual H-Bridge J2	EMS 1 A Dual H-Bridge J2	EMS 2 A Dual H-Bridge J1	Catatan
1	M2DIR1	M2D1	M2IN1	M2IN1	
2	M2DIR2	M2D2	M2IN2	M2IN2	
3	M2CS	NC	NC	M2CS	
4	M2PWM	M2PWM	M2EN	M2EN	
5	VCC	VCC	VCC	VCC	Karena pin ini terhubung, maka catu daya digital (+5V) untuk modul EMS H-Bridge **tidak** boleh diperoleh dari sumber lain
6	GND	PGND	PGND	PGND	Pin ini harus dihubungkan

Konektor H-BRIDGE 5A (J6 dan J7) merupakan merupakan **dua** set output untuk dihubungkan ke rangkaian *driver* motor DC atau motor stepper. **Konektor ini memiliki susunan dan letak yang kompatibel dengan modul EMS 5 A H-BRIDGE.**

J6			
Pin	**Nama**	**I/O**	**Fungsi**
1	M3DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-3** atau lilitan **stepper ke-2** (bersama-sama dengan M3DIR2) menggunakan modul EMS H-Bridge
2	M3DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-3** atau lilitan **stepper ke-2** (bersama-sama dengan M3DIR1) menggunakan modul EMS H-Bridge
3	GPIO1	O	Output digital ke-1
4	M3PWM	O	Output PWM ke motor DC 3
5	M3CS	I	Input tegangan analog ADC ke-3
6	GPIO2	O	Output digital ke-2
7	VCC	-	Tegangan output 5 Volt
8	GND	-	Titik referensi Ground
9	VCC	-	Tegangan output 5 Volt
10	GND	-	Titik referensi Ground

J7			
Pin	**Nama**	**I/O**	**Fungsi**
1	M4DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-4** atau lilitan **stepper ke-2** (bersama-sama dengan M4DIR2) menggunakan modul EMS H-Bridge
2	M4DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-4** atau lilitan **stepper ke-2** (bersama-sama dengan M4DIR1) menggunakan modul EMS H-Bridge
3	GPIO3	O	Output digital ke-3
4	M4PWM	O	Output PWM ke motor DC 4
5	M4CS	I	Input tegangan analog ADC ke-4
6	GPIO4	O	Output digital ke-4
7	VCC	-	Tegangan output 5 Volt
8	GND	-	Titik referensi Ground
9	VCC	-	Tegangan output 5 Volt
10	GND	-	Titik referensi Ground

Berikut ini contoh hubungan antara SPC MOTOR CONTROLLER dengan EMS 5 A H-BRIDGE (J6 & J7 masing-masing terhubung ke sebuah EMS 5 A H-BRIDGE):

Pin	**SPC MOTOR CONTROLLER J6**	**EMS 5 A H-Bridge J2**	**Catatan**
1	M3DIR1	MIN1	
2	M3DIR2	MIN2	
3	GPIO1	MSTAT1	
4	M3PWM	MEN	
5	M3CS	MCS	
6	GPIO2	MSLP	
7 dan 9	VCC	VCC	Jika catu daya digital (+5V) untuk modul EMS H-Bridge sudah diperoleh dari sumber lain, maka ke-2 pin ini tidak perlu dihubungkan
8 dan 10	GND	GND	Pin ini harus dihubungkan

Pin	**SPC MOTOR CONTROLLER J7**	**EMS 5 A H-Bridge J2**	**Catatan**
1	M4DIR1	MIN1	
2	M4DIR2	MIN2	
3	GPIO3	MSTAT1	
4	M4PWM	MEN	
5	M4CS	MCS	
6	GPIO4	MSLP	
7 dan 9	VCC	VCC	Jika catu daya digital (+5V) untuk modul EMS H-Bridge sudah diperoleh dari sumber lain, maka ke-2 pin ini tidak perlu dihubungkan
8 dan 10	GND	GND	Pin ini harus dihubungkan

Konektor H-BRIDGE 30A (J8 dan J9) merupakan merupakan **dua** set output untuk dihubungkan ke rangkaian *driver* motor DC atau motor stepper. **Konektor ini memiliki susunan dan letak yang kompatibel dengan modul EMS 30 A H-BRIDGE.**

J8			
Pin	**Nama**	**I/O**	**Fungsi**
1	M3DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-3** atau lilitan **stepper ke-2** (bersama-sama dengan M3DIR2) menggunakan modul EMS H-Bridge
2	M3DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-3** atau lilitan **stepper ke-2** (bersama-sama dengan M3DIR1) menggunakan modul EMS H-Bridge
3	GPIO1	O	Output digital ke-1
4	GPIO2	O	Output digital ke-2
5	M3CS	I	Input tegangan analog ADC ke-3
6	M3PWM	O	Output PWM ke motor DC 3
7	VCC	-	Tegangan output 5 Volt
8	GND	-	Titik referensi Ground
9	VCC	-	Tegangan output 5 Volt
10	GND	-	Titik referensi Ground

J9			
Pin	**Nama**	**I/O**	**Fungsi**
1	M4DIR1	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-4** atau lilitan **stepper ke-2** (bersama-sama dengan M4DIR2) menggunakan modul EMS H-Bridge
2	M4DIR2	O	Pin output untuk menentukan polaritas tegangan yang diberikan pada **motor DC ke-4** atau lilitan **stepper ke-2** (bersama-sama dengan M4DIR1) menggunakan modul EMS H-Bridge
3	GPIO3	O	Output digital ke-3
4	GPIO4	O	Output digital ke-4
5	M4CS	I	Input tegangan analog ADC ke-4
6	M4PWM	O	Output PWM ke motor DC 4
7	VCC	-	Tegangan output 5 Volt
8	GND	-	Titik referensi Ground
9	VCC	-	Tegangan output 5 Volt
10	GND	-	Titik referensi Ground

Berikut ini contoh hubungan antara SPC MOTOR CONTROLLER dengan EMS 30 A H-BRIDGE (J8 & J9 masing-masing terhubung ke sebuah EMS 30 A H-BRIDGE):

Pin	SPC MOTOR CONTROLLER J8	EMS 30 A H-Bridge J1	Catatan
1	M3DIR1	MIN1	
2	M3DIR2	MIN2	
3	GPIO1	MEN1	
4	GPIO2	MEN2	
5	M3CS	MCS	
6	M3PWM	MPWM	
7 dan 9	VCC	VCC	Jika catu daya digital (+5V) untuk modul EMS H-Bridge sudah diperoleh dari sumber lain, maka ke-2 pin ini tidak perlu dihubungkan
8 dan 10	GND	GND	Pin ini harus dihubungkan

Pin	SPC MOTOR CONTROLLER J9	EMS 30 A H-Bridge J1	Catatan
1	M4DIR1	MIN1	
2	M4DIR2	MIN2	
3	GPIO3	MEN1	
4	GPIO4	MEN2	
5	M4CS	MCS	
6	M4PWM	MPWM	
7 dan 9	VCC	VCC	Jika catu daya digital (+5V) untuk modul EMS H-Bridge sudah diperoleh dari sumber lain, maka ke-2 pin ini tidak perlu dihubungkan
8 dan 10	GND	GND	Pin ini harus dihubungkan

2.3. MEMILIH ENCODER YANG SESUAI

Pada bagian ini akan dijelaskan kiat-kiat yang dapat diikuti untuk memilih *encoder* sebagai sensor kecepatan yang sesuai dengan modul SPC MOTOR CONTROLLER. Pemilihan *encoder* dan pengaturan parameter kendali PID yang tepat akan menghasilkan kinerja yang optimal untuk kendali kecepatan motor dengan menggunakan modul SPC MOTOR CONTROLLER ini.

Modul SPC MOTOR CONTROLLER mampu mengendalikan 4 buah motor DC. Oleh karena itu, sebagai umpan balik untuk kendali PID, disediakan pula 4 jalur input *encoder* (1 untuk tiap motor). Tiap jalur input *encoder* tersebut mampu mencatat pulsa *encoder* sampai dengan frekuensi maksimum 10 kHz.

Pemilihan *encoder* yang sesuai berhubungan erat dengan parameter *sampling rate* yang ditentukan oleh pengguna. Tiap pulsa dari *encoder* akan dihitung dengan menggunakan *counter* pulsa. Kecepatan putaran motor DC diperoleh dengan membagi isi *counter* dengan perioda *sampling* yang ditentukan dan kemudian membagi lagi hasilnya dengan ketelitian *encoder* yang digunakan (parameter jumlah pulsa *encoder* tiap satu rotasi motor).

Ketelitian *encoder* minimum yang diperlukan dapat dihitung menggunakan persamaan berikut:

$$hole = \frac{minSpeed}{10 * (samplingPeriod)}$$

hole pada persamaan di atas merupakan ketelitian *encoder* minimum yang diperlukan. ***minSpeed*** adalah kecepatan minimum (dalam satuan RPS) yang ingin dibaca (sebagai contoh misalkan kita ingin mengukur kecepatan motor pada range 600 – 2400 RPM atau 10 – 40 RPS, maka kecepatan minimum yang ingin dibaca atau ***minSpeed*** adalah 10 RPS). Sedangkan ***samplingPeriod*** adalah perioda *sampling* (dalam satuan detik) yang telah diatur pada parameter PID (lihat **bagian 3.3.14**).

Sebagai ilustrasi, misalkan perioda *sampling* ditentukan sebesar 100 ms (0,1 detik) dan kita ingin mengukur kecepatan motor pada range 600 – 2400 RPM (10 – 40 RPS) dengan ketelitian 0,1 RPS. Ketelitian *encoder* minimum yang diperlukan adalah:

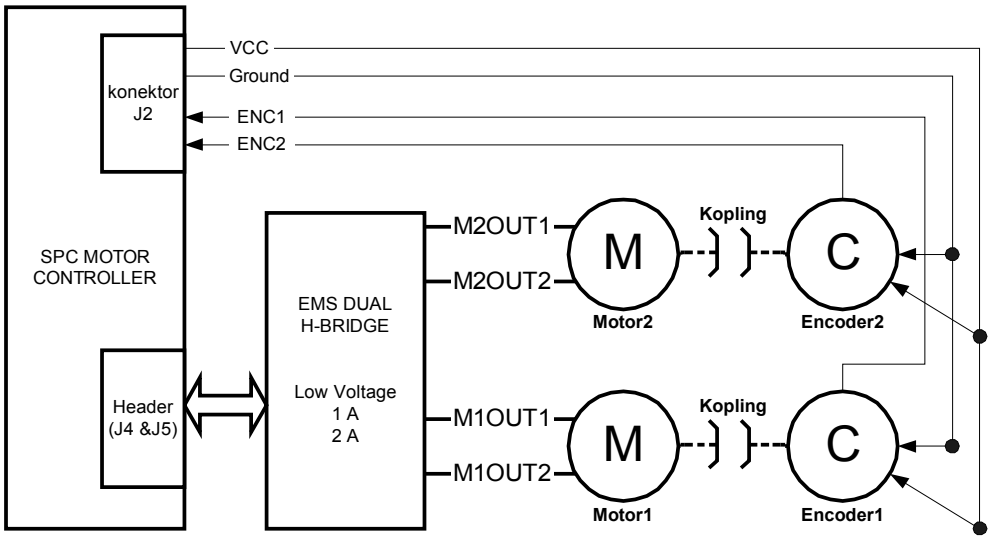
$$hole = \frac{10}{10 * (0.1)} = 10$$

Jadi dalam satu putaran motor, *encoder* minimum harus sudah menghasilkan 10 pulsa (atau dengan kata lain, gunakan *encoder* yang memiliki minimum 10 lubang).

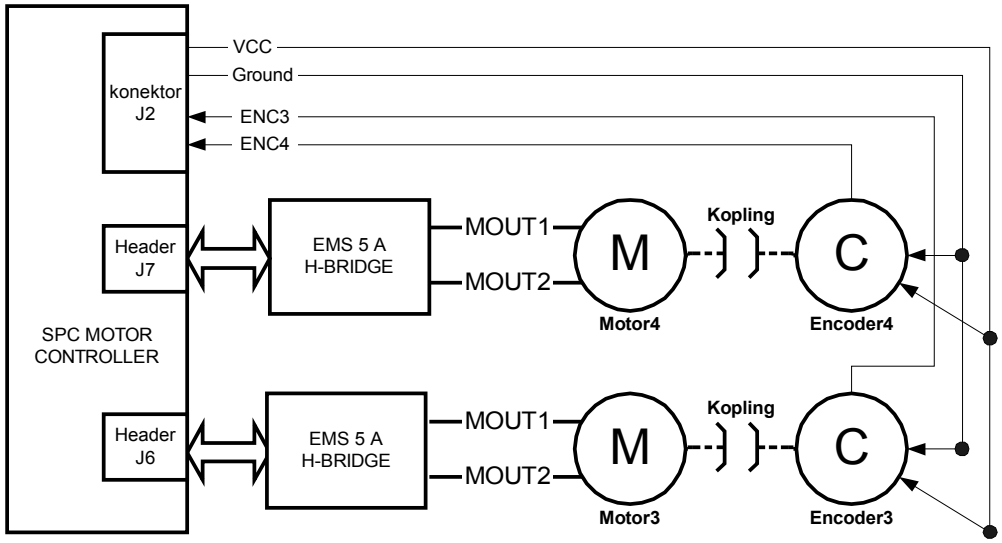
Jika lubang pada *encoder* lebih sedikit daripada yang dibutuhkan, maka gunakan perioda *sampling* yang lebih lama agar ketelitian 0,1 RPS dapat dicapai. Kemudian hitung lagi menggunakan persamaan di atas untuk menentukan apakah ketelitian *encoder* sudah mencukupi.

2.4. HUBUNGAN SPC MOTOR CONTROLLER DENGAN MOTOR DC DAN ENCODER

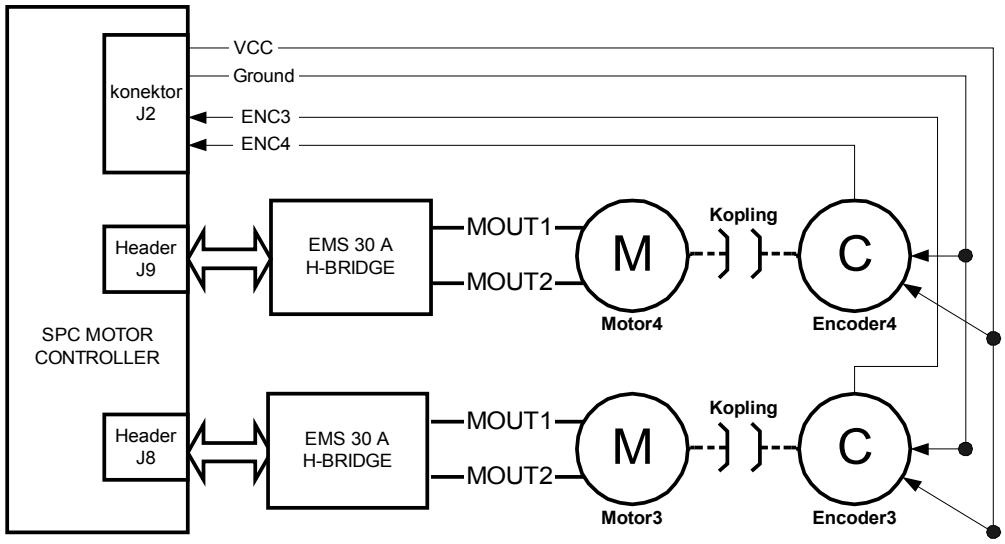
Berikut contoh hubungan antara motor DC, *encoder*, modul EMS LOW VOLTAGE DUAL H-BRIDGE, EMS 1 A DUAL H-BRIDGE, atau EMS 2 A DUAL H-BRIDGE dan modul SPC MOTOR CONTROLLER:



Berikut contoh hubungan antara motor DC, *encoder*, modul EMS 5 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:



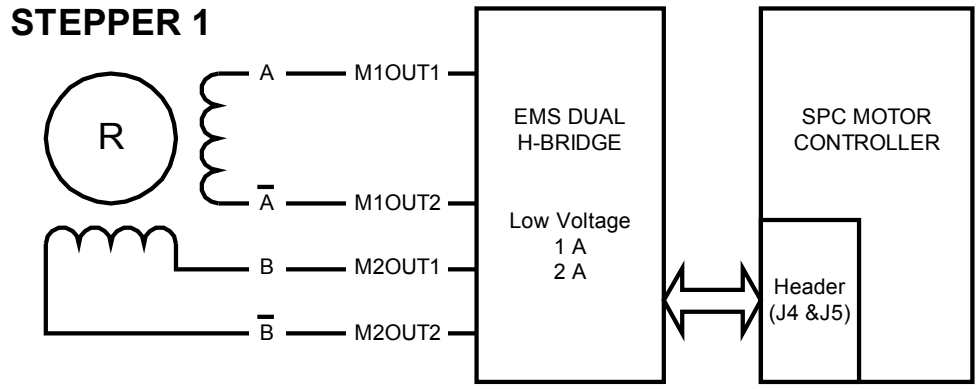
Berikut contoh hubungan antara motor DC, *encoder*, modul EMS 30 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:



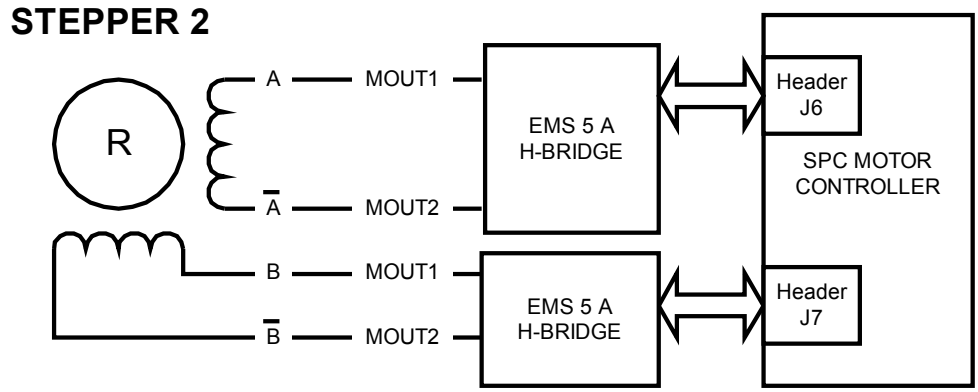
2.5. HUBUNGAN SPC MOTOR CONTROLLER DENGAN MOTOR STEPPER

2.5.1. MOTOR STEPPER BIPOLAR 4 KABEL

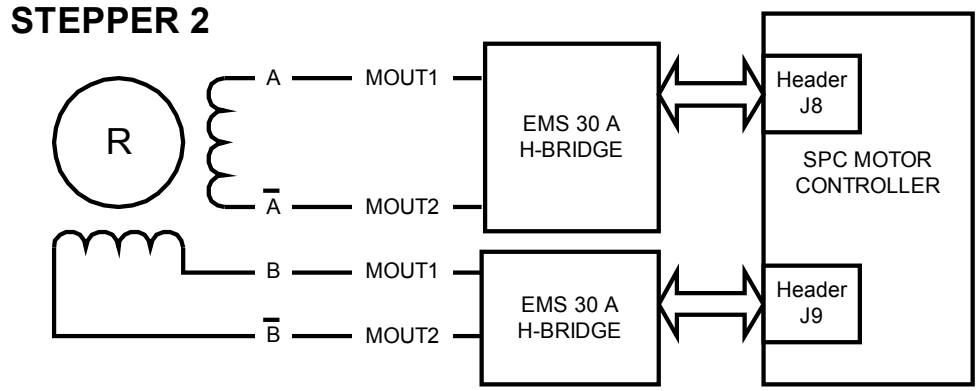
Berikut contoh hubungan antara motor stepper bipolar, modul EMS LOW VOLTAGE DUAL H-BRIDGE, EMS 1 A DUAL H-BRIDGE, atau EMS 2 A DUAL H-BRIDGE dan modul SPC MOTOR CONTROLLER:



Berikut contoh hubungan antara motor stepper bipolar, modul EMS 5 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:

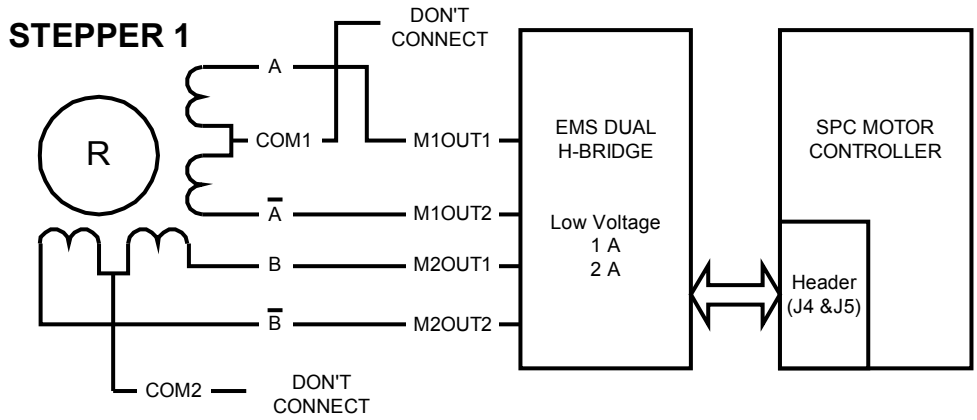


Berikut contoh hubungan antara motor stepper bipolar, modul EMS 30 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:

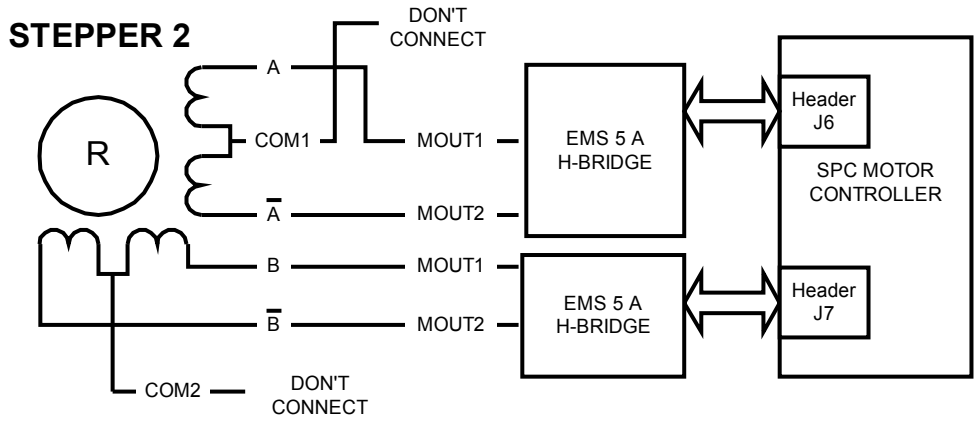


2.5.2. MOTOR STEPPER UNIPOLAR 6 KABEL

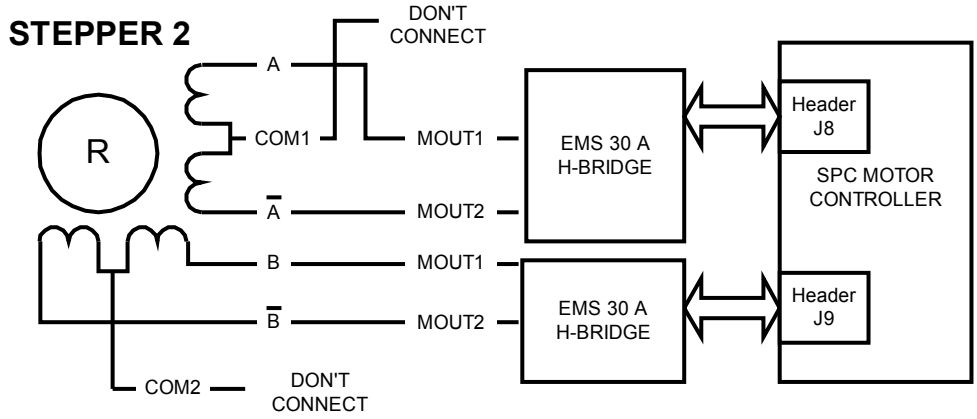
Berikut contoh hubungan antara motor stepper unipolar 6 kabel, modul EMS LOW VOLTAGE DUAL H-BRIDGE, EMS 1 A DUAL H-BRIDGE, atau EMS 2 A DUAL H-BRIDGE dan modul SPC MOTOR CONTROLLER:



Berikut contoh hubungan antara motor stepper unipolar 6 kabel, modul EMS 5 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:



Berikut contoh hubungan antara motor stepper unipolar 6 kabel, modul EMS 30 A H-BRIDGE dan modul SPC MOTOR CONTROLLER:



3. KOMUNIKASI DENGAN SPC MOTOR CONTROLLER

SPC MOTOR CONTROLLER memiliki antarmuka UART dan I^2C yang dapat digunakan untuk menerima perintah atau mengirim data.

3.1. ANTARMUKA UART

Parameter komunikasi UART adalah sebagai berikut: 38400 bps, 8 bit data, 1 bit stop, tanpa bit *parity*, dan tanpa *flow control*.

Semua perintah dan parameter dikirim menggunakan karakter ASCII (angka 123 dikirim dalam bentuk 3 karakter: "123", atau 3 byte ASCII: 31H 32H 33H). Karakter Spasi " " (20H) digunakan sebagai pemisah antar parameter perintah. Karakter \r (ASCII 13, *carriage return*) digunakan untuk mengakhiri transmisi. Untuk mengirimkan karakter \r pada HyperTerminal© dapat dilakukan dengan menekan tombol Enter pada *keyboard*.
Jika transmisi perintah berserta parameternya berhasil, maka SPC MOTOR CONTROLLER akan mengirimkan *string* "ACK". Jika perintah tidak dikenal, maka SPC MOTOR CONTROLLER akan mengirimkan *string* "NCK". Sedangkan jika perintah dikenal tetapi parameternya salah, maka SPC MOTOR CONTROLLER tidak akan mengirimkan balasan. Setelah mengirimkan "ACK", "NCK", ataupun data balasan, SPC MOTOR CONTROLLER akan selalu mengirimkan karakter \r (bilangan desimal 13 atau bilangan hexadesimal 0DH).

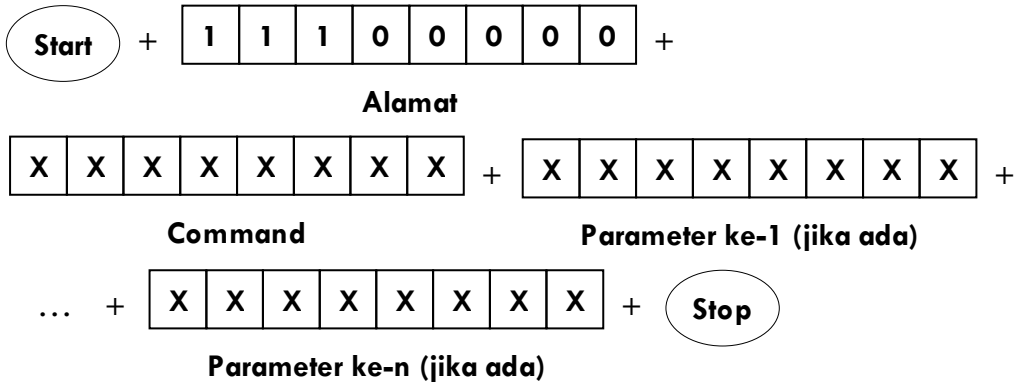
3.2. ANTARMUKA I^2C

SPC MOTOR CONTROLLER memiliki antarmuka I^2C. Pada antarmuka I^2C ini, modul SPC MOTOR CONTROLLER bertindak sebagai *slave* dengan alamat tetap E0H. Antarmuka I^2C pada SPC MOTOR CONTROLLER mendukung *bit rate* maksimum 100 kHz.

Semua perintah yang dikirim melalui antarmuka I^2C diawali dengan ***start condition*** dan kemudian diikuti dengan pengiriman 1 byte alamat tetap modul SPC MOTOR CONTROLLER (bilangan hexadesimal **E0H**). Setelah pengiriman alamat, selanjutnya *master* harus mengirim 1 byte data yang berisi **<command>** dan n byte data parameter perintah. Selanjutnya, setelah seluruh parameter perintah telah dikirim, urutan perintah diakhiri dengan ***stop condition***.

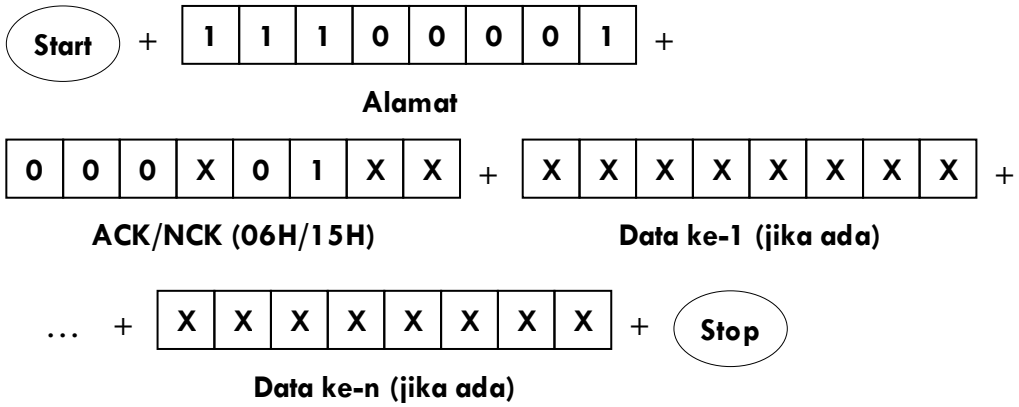
Pengiriman sebuah data parameter yang memiliki *range* lebih besar dari 255 desimal (lebih besar dari 1 byte) dikirim secara dua tahap. Satu byte data MSB dikirim lebih dahulu kemudian diikuti dengan data LSB. Misalnya parameter <maxSpeed> yang memiliki *range* 0 – 30000. Jika <maxSpeed> bernilai 100 maka byte MSB yang dikirim adalah 0 dan byte LSB yang dikirim adalah 100.

Berikut urutan yang harus dilakukan untuk mengirimkan perintah melalui antar muka I^2C. Perintah <**command**> dan parameter-parameternya yang bisa digunakan dapat dilihat pada **bagian 3.3**.



Jika transmisi perintah berserta parameternya berhasil, maka SPC MOTOR CONTROLLER akan menulis data hexadesimal **06H** pada *buffer* I²C-nya (*Acknowledged*). Sebaliknya jika perintah tidak dikenal atau parameter perintah salah, maka SPC MOTOR CONTROLLER akan menulis data hexadesimal **15H** pada *buffer* I²C-nya (*Not Acknowledged*). *Master* dapat mengirimkan perintah baca untuk membaca <**ACK**/**NCK**> tersebut. Jika perintah yang telah dikirimkan merupakan perintah yang meminta data ke modul SPC MOTOR CONTROLLER, maka data-data tersebut dapat dibaca langsung setelah membaca *acknowledgement*. Jika ada perintah baru yang diterima, maka *buffer* I²C akan dibersihkan (walaupun data *acknowledgement* tidak dibaca oleh *master*).

Berikut urutan yang harus dilakukan untuk membaca *acknowledgement* dan/atau data lain.



3.3. DAFTAR PERINTAH UNTUK ANTARMUKA UART dan I²C

Berikut ini adalah daftar perintah dan parameter SPC MOTOR CONTROLLER. Beberapa parameter juga akan disimpan dalam EEPROM modul SPC MOTOR CONTROLLER. Parameter tersebut akan dibaca saat SPC MOTOR CONTROLLER baru dinyalakan.

3.3.1. PING

Fungsi	Untuk menguji koneksi UART dan memastikan bahwa SPC MOTOR CONTROLLER siap menerima perintah
Command UART	**PG**\r
Command I²C	**00H**
Parameter	-
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	-

Contoh dengan antarmuka UART:
User : `PG\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I^2C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x00);     // Perintah "Ping"
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp = i2c_read(0); // Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.2. GET VERSION NUMBER

Fungsi	Untuk mengetahui versi perangkat lunak yang tertanam pada modul SPC MOTOR CONTROLLER
Command UART	**VR**\r
Command I^2C	**01H**
Parameter	-
Respon UART	ACK\r <string>\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I^2C	06H <version> <revision> → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Setelah mengirimkan respon UART, SPC MOTOR CONTROLLER akan mengirimkan <string> yang menyatakan versi perangkat lunak yang tertanam pada modul SPC MOTOR CONTROLLER. • Setelah mengirimkan respon I^2C, SPC MOTOR CONTROLLER akan mengirimkan 2 byte data heksadesimal yang menyatakan nomor versi <version> dan nomor revisi <revision>.

Contoh dengan antarmuka UART:
User : `VR\r`
SPC : `ACK\r`
`SPCMotorControl v0.1\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I^2C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x01);     // Perintah "Get Version Number"
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
```

```
i2c_write(0xE1);    // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(1);// Data Acknowledgement
temp2 = i2c_read(1);// Version Number
temp3 = i2c_read(0);// Revision Number
i2c_stop();         // Stop Condition
```

3.3.3. SET MODE

Fungsi	Untuk menentukan mode operasi SPC MOTOR CONTROLLER
Command UART	**MD** <Mode>\r
Command I²C	**02H** <Mode>
Parameter	<Mode> 0 → Motor DC C*ontroller* 1 → Motor Stepper C*ontroller*
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	4 ms
Keterangan	Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengatur SPC MOTOR CONTROLLER sebagai pengendali motor DC:
User : `MD 0\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();        // Start Condition
i2c_write(0xE0);    // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x02);    // Perintah "Set Mode"
i2c_write(0x00);    // Pilih Mode "Motor DC Controller"
i2c_stop();         // Stop Condition

delay_ms(4);        // delay 4 ms

i2c_start();        // Start Condition
i2c_write(0xE1);    // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();         // Stop Condition
```

3.3.4. AUTO RUN

Fungsi	Untuk memutar motor secara terus menerus sesuai dengan pengaturan yang telah ditentukan sebelumnya
Command UART	**RN** <Motor number> <on/off>\r
Command I²C	**03H** <Motor number> <on/off>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <on/off> 0 → off (*autorun disabled*) 1 → on (*autorun enabled*) Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <on/off> 0 → off (*autorun disabled*) 1 → on (*autorun enabled*) 2 → off (*autorun disabled + soft reset*)
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Jika mode autorun diaktifkan (*enabled*), maka motor akan bergerak sesuai dengan arah dan *setpoint* kecepatan yang telah ditentukan sebelumnya. • Pada saat *soft reset*, posisi motor stepper sekarang akan ditentukan menjadi posisi *home* (posisi absolut 0°), seluruh arus dan tegangan yang dialirkan ke kumparan motor akan diputus. • Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengaktifkan autorun motor DC yang terhubung pada output ke-3 (mode operasi telah diatur pada mode motor DC *controller*):
User : `RN 2 1\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I^2C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x03);     // Perintah "Auto Run"
i2c_write(0x03);     // Untuk Motor DC ke-3
i2c_write(0x01);     // Auto Run Enable
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.5. SET MOTOR PARAMETER

Fungsi	Untuk menentukan parameter motor
Command UART	Pada **mode motor DC** *controller* : **PR** <Motor number> <max speed> <pulPerRot>\r Pada **mode motor stepper** *controller* : **PR** <Motor number> <degPerStep> <max pps>\r
Command I^2C	Pada **mode motor DC** *controller* : **04H** <Motor number> <max speed> <pulPerRot> Pada **mode motor stepper** *controller* : **04H** <Motor number> <degPerStep> <max pps>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <max speed> 0 – 30000 → kecepatan maksimum motor DC yang digunakan (dalam satuan 0,1 RPS). *Range* yang diperbolehkan 0 – 3000,0 RPS. <pulPerRot> 1 – 30000 → banyaknya pulsa tiap satu rotasi motor untuk encoder yang digunakan. *Range* yang diperbolehkan 1 – 30000 pulsa per rotasi.

Parameter	Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <degPerStep> 1 – 150 → ketelitian derajat per step pada motor stepper yang digunakan (dalam satuan 0,1°). *Range* yang diperbolehkan 0,1° – 15,0°. <max PPS> 1 – 255 → kecepatan *stepping* maksimum motor stepper yang digunakan. *Range* diperbolehkan 1 – 255 PPS.
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	4 ms
Keterangan	• Parameter-parameter motor tersebut digunakan sebagai informasi untuk mengatur batas-batas operasi modul SPC MOTOR CONTROLLER. • Jika perintah ini diterima, maka parameter motor tersebut juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART jika menggunakan motor DC dengan kecepatan maksimum 2400 RPM (40,0 RPS) yang terhubung pada output ke-1 dan *encoder*-nya memiliki ketelitian 116 pulsa tiap satu rotasi motor (mode operasi telah diatur pada mode motor DC *controller*):

User : `PR 0 400 116\r`

SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x04);     // Perintah "Set Motor Parameter"
i2c_write(0x00);     // Untuk Motor DC ke-1
i2c_write(0x01);     // Max Speed MSB (400 desimal = 0x0190)
i2c_write(0x90);     // Max Speed LSB
i2c_write(0x00);     // PulPerRot MSB (116 desimal = 0x0074)
i2c_write(0x74);     // PulPerRot LSB
i2c_stop();          // Stop Condition

delay_ms(4);         // delay 4 ms

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.6. SET MOTOR DIRECTION

Fungsi	Untuk menentukan arah motor
Command UART	**DR** <Motor number> <cw/ccw>\r
Command I²C	**05H** <Motor number> <cw/ccw>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <cw/ccw> 0 → motor berputar searah jarum jam 1 → motor berputar berlawanan arah jarum jam Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <cw/ccw> 0 → motor berputar searah jarum jam 1 → motor berputar berlawanan arah jarum jam
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengatur motor stepper yang dihubungkan pada output ke-2 berputar searah jarum jam (mode operasi telah diatur pada mode motor stepper *controller*):
User : `DIR 1 0\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x05);     // Perintah "Set Motor Direction"
i2c_write(0x01);     // Untuk Motor Stepper ke-2
i2c_write(0x00);     // Arah putaran motor
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
```

```
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();         // Stop Condition
```

3.3.7. SET STEP TYPE

Fungsi	Untuk menentukan tipe *step* untuk motor stepper
Command UART	**TP** <Motor number> <stepType>\r
Command I²C	**06H** <Motor number> <stepType>
Parameter	Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <stepType> 1 → FullStep 2 → HalfStep 4 → MicroStep 4 8 → MicroStep 8 16 → MicroStep 16 32 → MicroStep 32
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Jika penggatian tipe *step* dilakukan, maka posisi *home* motor stepper sebaiknya ditentukan ulang. • Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengatur agar motor stepper yang dihubungkan pada output ke-2 bergerak menggunakan tipe HalfStep (mode operasi telah diatur pada mode motor stepper *controller*):
User : `TP 1 2\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();        // Start Condition
i2c_write(0xE0);    // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x06);    // Perintah "Set Step Type"
i2c_write(0x01);    // Untuk Motor Stepper ke-2
i2c_write(0x02);    // Menggunakan tipe "HalfStep"
i2c_stop();         // Stop Condition

delay_us(10);       // delay 10 us

i2c_start();        // Start Condition
i2c_write(0xE1);    // Baca ke modul SPC MOTOR CONTROLLER
```

```
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();         // Stop Condition
```

3.3.8. SET STEP RATE

Fungsi	Untuk menentukan kecepatan *step* untuk motor stepper
Command UART	**SR** <Motor number> <step rate>\r
Command I²C	**07H** <Motor number> <step rate>
Parameter	Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <step rate> *5* – 2550 → kecepatan *stepping* motor stepper (dalam satuan 0,1 step per detik). *Range* diperbolehkan 0,*5* – 255,0 step per detik. Jika pengaturan kecepatan lebih besar dari parameter **max PPS** yang telah tersimpan, maka kecepatan *stepping* motor stepper menjadi sama dengan **max PPS.**
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	4 ms
Keterangan	Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengatur agar motor stepper yang dihubungkan pada output ke-1 bergerak dengan kecepatan 50 step per detik (mode operasi telah diatur pada mode motor stepper *controller*):
User : `SR 0 500\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();        // Start Condition
i2c_write(0xE0);    // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x07);    // Perintah "Set Step Rate"
i2c_write(0x00);    // Untuk Motor Stepper ke-1
i2c_write(0x01);    // Step Rate MSB (500 desimal = 0x01F4)
i2c_write(0xF4);    // Step Rate LSB
i2c_stop();         // Stop Condition

delay_ms(4);        // delay 4 ms

i2c_start();        // Start Condition
i2c_write(0xE1);    // Baca ke modul SPC MOTOR CONTROLLER
```

```
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();         // Stop Condition
```

3.3.9. SET HOME POSITION

Fungsi	Untuk menentukan posisi motor stepper sekarang menjadi posisi *Home*
Command UART	**HM** <Motor number>\r
Command I²C	**08H** <Motor number>
Parameter	Pada **mode motor stepper** controller. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Perintah ini **tidak** tersimpan di EEPROM. Posisi motor stepper pada saat SPC MOTOR CONTROLLER baru dinyalakan akan menjadi posisi *Home*). • Jika perintah ini diberikan, maka posisi stepper sekarang akan ditentukan menjadi posisi *Home* (posisi absolut 0°).

Contoh dengan antarmuka UART untuk menjadikan posisi motor stepper yang dihubungkan pada output ke-1 menjadi posisi *Home* atau posisi absolut 0° (mode operasi telah diatur pada mode motor stepper *controller*):
User : `HM 0\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x08);     // Perintah "Set Home Position"
i2c_write(0x00);     // Untuk Motor Stepper ke-1
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.10. GOTO POSITION

Fungsi	Untuk memutar motor stepper ke posisi absolut tertentu
Command UART	**GT** <Motor number> <position>\r
Command I²C	**09H** <Motor number> <position>
Parameter	Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <position> 0 – 3599 → Posisi absolut motor stepper (dalam satuan 0,1°). *Range* diperbolehkan 0° – 359,9°.
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	4 ms
Keterangan	• Arah gerakan motor stepper mengikuti arah gerak yang sudah ditentukan (perintah Set Motor Direction). • Keakuratan posisi motor stepper tergantung pada parameter derajat per step motor stepper dan tipe *step* yang digunakan. • Jika perintah ini diterima, maka **mode *autorun* akan otomatis menjadi *disabled*** dan kondisinya disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengendalikan motor stepper yang dihubungkan pada output ke-2 bergerak ke posisi absolut 180,5° (mode operasi telah diatur pada mode motor stepper *controller*):
User : `GT 1 1805\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x09);     // Perintah "Goto Position"
i2c_write(0x01);     // Untuk Motor Stepper ke-2
i2c_write(0x07);     // Position MSB (1805 desimal = 0x070D)
i2c_write(0x0D);     // Position LSB
i2c_stop();          // Stop Condition

delay_ms(4);         // delay 4 ms

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.11. MOVE X DEGREE

Fungsi	Untuk memutar motor stepper relatif terhadap posisi sekarang
Command UART	**MV** <Motor number> <relative degree>\r
Command I²C	**0AH** <Motor number> <relative degree>
Parameter	Pada **mode motor stepper** *controller*. <Motor number> 0 → motor stepper yang terhubung pada output ke-1 1 → motor stepper yang terhubung pada output ke-2 <relative degree> 0 – 3599 → berapa derajat motor stepper harus bergerak dari posisi sekarang (dalam satuan 0,1°). *Range* yang diperbolehkan 0° – 359,9°.
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	4 ms
Keterangan	• Arah gerakan motor stepper mengikuti arah gerak yang sudah ditentukan (perintah Set Motor Direction). • Keakuratan posisi motor stepper tergantung pada parameter derajat per step motor stepper dan tipe *step* yang digunakan. • Jika perintah ini diterima, maka **mode *autorun* akan otomatis menjadi *disabled*** dan kondisinya disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengendalikan motor stepper yang dihubungkan pada output ke-2 bergerak 45,0° dari posisi sekarang (mode operasi telah diatur pada mode motor stepper *controller*):
User : `MV 1 450\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0A);     // Perintah "Move X Degree"
i2c_write(0x01);     // Untuk Motor Stepper ke-2
i2c_write(0x01);     // Degree MSB (450 desimal = 0x01C2)
i2c_write(0xC2);     // Degree LSB
i2c_stop();          // Stop Condition

delay_ms(4);         // delay 4 ms

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
```

```
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();         // Stop Condition
```

3.3.12. SET SETPOINT

Fungsi	Untuk menentukan *setpoint* kecepatan yang diinginkan
Command UART	**SP** <Motor number> <setpoint>\r
Command I²C	**0BH** <Motor number> <setpoint>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <setpoint> 0 – 30000 → *setpoint* kecepatan motor DC (dalam satuan 0,1 RPS). *Range* yang diperbolehkan 0 – 3000,0 RPS.
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	Jika perintah ini diterima, maka parameternya juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengendalikan motor DC yang dihubungkan pada output ke-3 bergerak dengan kecepatan 1200 RPM atau 20,0 RPS (mode operasi telah diatur pada mode motor DC c*ontroller*):
User : `SP 2 200\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0B);     // Perintah "Set Setpoint"
i2c_write(0x02);     // Untuk Motor DC ke-3
i2c_write(0x00);     // Setpoint MSB (200 desimal = 0x00C8)
i2c_write(0xC8);     // Setpoint LSB
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.13. GET SPEED

Fungsi	Untuk mengetahui kecepatan motor DC
Command UART	**GS** <Motor number>\r
Command I²C	**0CH** <Motor number>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4
Respon UART	ACK\r <speed>\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H <speedH> <speedL> → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Setelah mengirimkan respon UART, SPC MOTOR CONTROLLER akan mengirimkan 5 data *string* <speed> yang menyatakan kecepatan motor DC. Kecepatan motor DC hasil pembacaan adalah <speed>/10 RPS. • Setelah mengirimkan respon I²C, SPC MOTOR CONTROLLER akan mengirimkan 2 byte data heksadesimal <speedH> dan <speedL> yang menyatakan data kecepatan motor DC. Kecepatan motor DC = <speedH> * 256 + <speedL> (dalam satuan RPS).

Contoh dengan antarmuka UART untuk mengetahui kecepatan motor DC yang dihubungkan pada output ke-1 (mode operasi telah diatur pada mode motor DC *controller*):

User : `GS 0\r`

SPC : `ACK\r`

`00253\r`

Kecepatan motor DC hasil pembacaan adalah 25,3 RPS atau 1518 RPM.

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0C);     // Perintah "Get Speed"
i2c_write(0x00);     // Untuk Motor DC ke-1
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(1);// Data Acknowledgement
temp2 = i2c_read(1);// speed MSB
```

```
temp3 = i2c_read(0);// speed LSB
i2c_stop();         // Stop Condition
```

Kecepatan motor DC = temp2 * 256 + temp3 (dalam satuan RPS).

3.3.14. SET PID PARAMETER

Fungsi	Untuk menentukan parameter PID
Command UART	**PI** <Motor number> <P> <I> <D> <tol> <T>\r
Command I²C	**0DH** <Motor number> <P> <I> <D> <tol> <T>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <P> 1 – 30000 → Konstanta Proporsional Kp = 100/P <I> 1 – 30000 → Konstanta Integral Ki = 10/I <D> 1 – 30000 → Konstanta Diferensial Kd = 1000/D <tol> 0 – 30000 → Toleransi kesalahan keadaan tunak (dalam satuan 0,1 RPS). *Range* yang diperbolehkan 0 – 3000,0 RPS. <T> 2 – 30000 → Perioda *sampling*. *Range* yang diperbolehkan 2 ms – 30000 ms (500 Hz – 1/30 Hz).
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	27 ms
Keterangan	• Parameter-parameter PID tersebut digunakan sebagai informasi untuk algoritma kendali PID modul SPC MOTOR CONTROLLER. • Jika perintah ini diterima, maka parameter-parameter PID tersebut juga akan disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk menala parameter kendali PID untuk motor DC yang terhubung pada output ke-1 sehingga konstanta proporsional Kp = 10 (100/**10**), konstanta integral Ki = 0,001 (10/**10000**), konstanta diferensial Kd = 5 (1000/**200**) , kecepatan *sampling* kendali 100 Hz (perioda *sampling* **10** ms), dan toleransi kesalahan keadaan tunak 0,5 RPS atau 30 RPM (mode operasi telah diatur pada mode motor DC *controller*):

User : `PI 0 10 10000 200 5 10\r`

SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0D);     // Perintah "Set PID Parameter"
i2c_write(0x00);     // Untuk Motor DC ke-1

i2c_write(0x00);     // Kp MSB (10 desimal = 0x000A)
i2c_write(0x0A);     // Kp LSB

i2c_write(0x27);     // Ki MSB (10000 desimal = 0x2710)
i2c_write(0x10);     // Ki LSB

i2c_write(0x00);     // Kd MSB (200 desimal = 0x00C8)
i2c_write(0xC8);     // Kd LSB

i2c_write(0x05);     // Toleransi Error

i2c_write(0x00);     // Sampling MSB (10 desimal = 0x000A)
i2c_write(0x0A);     // Sampling LSB

i2c_stop();          // Stop Condition

delay_ms(27);        // delay 27 ms

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.15. SET PWM

Fungsi	Untuk menentukan *duty cycle* PWM motor DC
Command UART	**PM** <Motor number> <pwm level>\r
Command I²C	**0EH** <Motor number> <pwm level>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4 <pwm level> 0 – 255 → persentase *duty cycle* yang diberikan (0 = 0%; 255 = 100%)
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs

Keterangan	• Nilai PWM **tidak** akan disimpan di EEPROM. Saat modul SPC baru dinyalakan, nilai PWM adalah 0 (nol). • Jika perintah ini diterima, maka **mode *autorun* akan otomatis menjadi *disabled*** dan kondisinya disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengendalikan kecepatan motor DC dengan mengatur sendiri nilai PWM yang diberikan untuk motor DC yang terhubung pada output ke-2 (mode operasi telah diatur pada mode motor DC *controller*). Misalkan *duty cycle* yang diinginkan 50% (0,5 * 255 = 128):
User : `PM 1 128\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I^2C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0E);     // Perintah "Set PWM"
i2c_write(0x01);     // Untuk Motor DC ke-2
i2c_write(128);      // Level PWM
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.16. BRAKE

Fungsi	Untuk mengaktifkan kondisi *brake* motor DC
Command UART	**BR** <Motor number>\r
Command I^2C	**0FH** <Motor number>
Parameter	Pada **mode motor DC** *controller*. <Motor number> 0 → motor DC yang terhubung pada output ke-1 1 → motor DC yang terhubung pada output ke-2 2 → motor DC yang terhubung pada output ke-3 3 → motor DC yang terhubung pada output ke-4
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I^2C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs

Keterangan	• Bila modul SPC MOTOR CONTROLLER dihubungkan dengan modul EMS H-Bridge, maka jika perintah ini diterima, maka motor DC akan *brake* (kedua kabel input tegangan motor DC terhubung ke ground). • Kondisi brake **tidak** akan disimpan di EEPROM. Bila modul SPC MOTOR CONTROLLER dihubungkan dengan modul EMS H-Bridge, maka saat modul SPC baru dinyalakan, kondisi motor DC akan berada pada kondisi *free run/free wheel* (kedua input tegangan motor DC akan berada pada keadaan *high impedance*). • Jika perintah ini diterima, maka **mode *autorun* akan otomatis menjadi *disabled*** dan kondisinya disimpan pada EEPROM modul SPC MOTOR CONTROLLER.

Contoh dengan antarmuka UART untuk mengaktifkan kondisi *brake* untuk motor DC yang terhubung pada output ke-2 (mode operasi telah diatur pada mode motor DC *controller*):
User : `BR 1\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I^2C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x0F);     // Perintah "Brake"
i2c_write(0x01);     // Untuk Motor DC ke-2
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

3.3.17. GET ADC

Fungsi	Untuk mengetahui nilai ADC 8-bit
Command UART	**AD** <ADC channel>\r
Command I^2C	**10H** <ADC channel>
Parameter	<ADC channel> 0 – 3 → Kanal ADC
Respon UART	ACK\r <value>\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I^2C	06H <value> → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs

Keterangan	• Setelah mengirimkan respon UART, SPC MOTOR CONTROLLER akan mengirimkan 3 byte data <value> yang menyatakan hasil konversi ADC dalam bentuk *string*. • Setelah mengirimkan respon I²C, SPC MOTOR CONTROLLER akan mengirimkan 1 byte data <value> yang menyatakan hasil konversi ADC dalam bentuk heksadesimal. • Nilai tegangan yang dibaca = <value> / 256 * 5 Volt. • Modul SPC MOTOR CONTROLLER memiliki 4 kanal ADC 8-bit dengan ketelitian ±0,5 LSB. ADC ini bisa digunakan pengguna untuk berbagai keperluan seperti misalnya sensor arus, suhu, atau input analog lainnya. ADC ini menggunakan tegangan referensi 5 Volt dan maksimum *sampling rate* 25 kHz.

Contoh dengan antarmuka UART untuk membaca hasil konversi tegangan analog yang terhubung ke kanal 0:

User : `AD 0\r`
SPC : `ACK\r`
`128\r`

Hasil konversi adalah 128 atau (128 / 256) * 5 Volt = 2,5 Volt.

Berikut ini contoh pseudo code C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x10);     // Perintah "Get ADC"
i2c_write(0x00);     // Kanal ADC ke-1
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us

i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(1);// Data Acknowledgement
temp2 = i2c_read(0);// Data hasil konversi ADC
i2c_stop();          // Stop Condition
```

Hasil konversi = (temp2 / 256) * 5 Volt.

3.3.18. SET GP I/O

Fungsi	Untuk menentukan logika pada pin GP I/O
Command UART	**IO** <I/O number> <High/Low>\r
Command I²C	**11H** <I/O number> <High/Low>
Parameter	<I/O number> 0 → GP I/O 1 1 → GP I/O 2 2 → GP I/O 3 3 → GP I/O 4 <High/Low> 0 → Low 0V (*sink*) 1 → High 5V (*internal pull-up resistor* diaktifkan) (***default* saat modul SPC baru dinyalakan**)
Respon UART	ACK\r → jika perintah dikenali NCK\r → jika perintah tidak dikenali
Respon I²C	06H → jika perintah dikenali 15H → jika perintah tidak dikenali
Delay antara Command dan Respon	10 µs
Keterangan	• Pin I/O ini dapat digunakan untuk mengatur fungsi-fungsi tambahan yang ada pada modul EMS 5 A H-BRIDGE dan 30 A H-BRIDGE. • Logika High pada pin I/O tidak bersifat Output High (s*ource*), tetapi merupakan Input yang *internal pull-up resistor*-nya diaktifkan (masih bisa ditarik menjadi Low oleh *external device*). Hal tersebut karena beberapa pin pada modul EMS H-Bridge ada yang dapat berfungsi sebagai input (*enable/disable* h-bridge) atau sebagai output (jika terjadi kondisi *fault* pada h-bridge akan menjadi Low). • Kondisi pin **tidak** akan disimpan di EEPROM. Saat modul SPC MOTOR CONTROLLER baru dinyalakan, kondisi pin akan berlogika **High**.

Contoh dengan antarmuka UART untuk membuat agar logika GP I/O 2 menjadi Low:
User : `IO 1 0\r`
SPC : `ACK\r`

Berikut ini contoh *pseudo code* C untuk menggunakan perintah ini dengan antarmuka I²C:

```
i2c_start();         // Start Condition
i2c_write(0xE0);     // Tulis ke modul SPC MOTOR CONTROLLER
i2c_write(0x11);     // Perintah "Set GP I/O"
i2c_write(0x01);     // Port I/O ke-2
i2c_write(0x00);     // Set State Low
i2c_stop();          // Stop Condition

delay_us(10);        // delay 10 us
```

```
i2c_start();         // Start Condition
i2c_write(0xE1);     // Baca ke modul SPC MOTOR CONTROLLER
temp1 = i2c_read(0);// Data Acknowledgement
i2c_stop();          // Stop Condition
```

4. ANTARMUKA LEBAR PULSA UNTUK KENDALI MOTOR DC

Jika modul SPC MOTOR CONTROLLER digunakan untuk mengendalikan motor DC, selain antarmuka UART dan IC, terdapat antarmuka lebar pulsa yang dapat digunakan untuk mengendalikan kecepatan dan arah putaran motor DC.
Pada modul SPC MOTOR CONTROLLER terdapat 4 jalur input lebar pulsa, dimana tiap jalur dapat mengendalikan 1 motor DC.

Tiap jalur dapat menerima pulsa (pulsa positif / logika High) dengan lebar 2 ms – 22 ms, dengan frekuensi maksimum 40 Hz. Motor akan berhenti jika pulsa yang diberikan memiliki lebar 12 ms. Jika pulsa yang diberikan (misal: pada jalur 1) memiliki lebar 2 ms, maka motor DC (yang terhubung pada output ke-1) akan dikendalikan agar berputar pada kecepatan maksimumnya searah dengan jarum jam (dengan kendali PID). Sebaliknya, jika pulsa yang diberikan memiliki lebar 22 ms, maka motor DC (yang terhubung pada output ke-1) akan dikendalikan agar berputar pada kecepatan maksimumnya berlawanan arah dengan jarum jam.

Mulai lebar pulsa 2 ms sampai dengan 12 ms, *setpoint* kecepatan akan berubah secara linier mulai dari kecepatan maksimum sampai dengan berhenti total. Demikian juga, mulai lebar pulsa 12 ms sampai dengan 22 ms, *setpoint* kecepatan akan berubah secara linier mulai dari berhenti total sampai dengan kecepatan maksimum, tetapi pada arah putaran motor sebaliknya.

Kecepatan maksimum putaran motor diatur melalui parameter <max speed> melalui antarmuka yang lain (lihat **bagian 3.3.5**).

Untuk lebar pulsa antara 2 ms sampai dengan 12 ms, *setpoint* kecepatan yang diberikan ke kendali motor sebesar:

$$setpoint = maxSpeed - (\frac{lebarPulsa - 2\,ms}{10\,ms} * maxSpeed\ )$$

Untuk lebar pulsa antara 12 ms sampai dengan 22 ms, *setpoint* kecepatan yang diberikan ke kendali motor sebesar :

$$setpoint = maxSpeed - (\frac{22\,ms - lebarPulsa}{10\,ms} * maxSpeed\ )$$

Modul SPC MOTOR CONTROLLER mampu menangkap perubahan lebar pulsa sebesar 40 µs. Jadi mulai lebar pulsa 2 ms sampai dengan 22 ms, total dibagi menjadi 500 level perubahan *setpoint*.

Sebagai ilustrasi, misalkan kecepatan maksimum motor DC ke-1 telah diatur sebesar 1500 RPM (25 RPS). Maka jika pada jalur-1 antarmuka lebar pulsa kita berikan pulsa sebesar 2 ms, maka motor DC ke-1 akan berputar dengan kecepatan 1500 RPM (25 RPS). Jika pulsa yang diberikan sebesar 10 ms, maka motor DC ke-1 akan berputar dengan kecepatan 300 RPM (5 RPS).

Untuk mengurangi beban kerja penghasil pulsa (misalkan pulsa tersebut dihasilkan dari sistem barbasis mikroprosesor atau mikrokontroler yang lain), pada antarmuka lebar pulsa ini disediakan fasilitas ***latching* pulsa**. Jika setelah mengirimkan pulsa jalur input lebar pulsa diberi logika Low terus menerus, maka *setpoint* dan arah putaran motor yang diatur melalui pulsa sebelumnya akan tetap tersimpan dan dijalankan oleh sistem kendali PID. Sebaliknya, jika pengguna ingin menghentikan kerja motor (tanpa terus menerus mengirimkan pulsa selebar 12 ms), maka pengguna dapat memberikan logika High (minimal 25 ms) pada jalur input tersebut (logika *default* saat jalur input tidak terhubung kemana-mana adalah High).

5. PROSEDUR PENGUJIAN

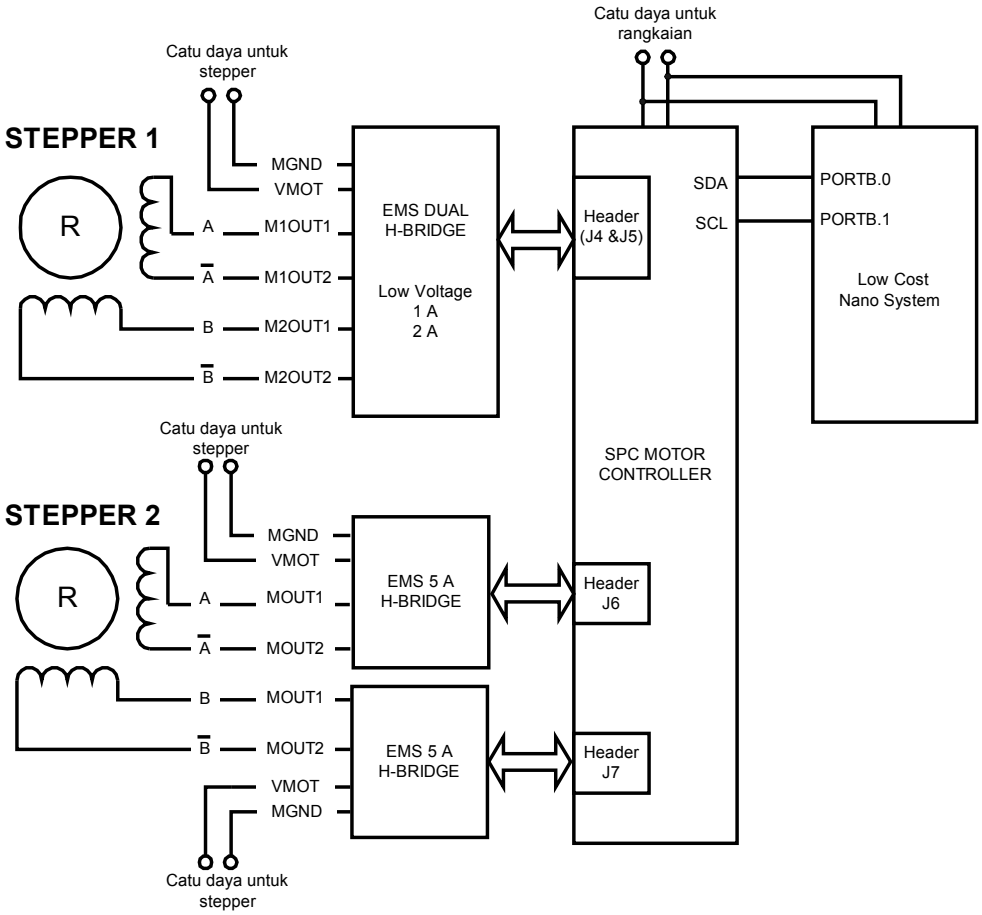
1. Atur *jumper* J12 dan J13 ke posisi 1-2 agar antarmuka UART menggunakan level tegangan RS-232.
2. Hubungkan sumber catu daya pada modul SPC MOTOR CONTROLLER.
3. Jika catu daya telah terhubung dengan baik, maka LED akan menyala.
4. Hubungkan konektor RJ11 (J11) dengan *port* serial komputer dengan menggunakan kabel serial yang disertakan.
5. Jalankan perangkat lunak komunikasi serial pada komputer (misalnya: HyperTerminal, dsb.).
6. Untuk menguji koneksi UART kirimkan perintah **PING\r (**ketikkan **PING** lalu tekan tombol **Enter)**. Jika koneksi antarmuka UART bekerja dengan baik, maka modul SPC MOTOR CONTROLLER akan membalas dengan memberikan *acknowledgement* **ACK\r** atau **ACK** (lihat **bagian 3.3.1**).
7. Lakukan pengujian pin-pin pengaturan arah untuk motor ke-1 (M1DIR1 dan M1DIR2) dengan memberikan perintah **SET MOTOR DIRECTION** motor ke-1 agar berputar searah jarum jam. Jika pin-pin pengaturan arah bekerja dengan baik, maka M1DIR1 akan memiliki logika HIGH dan M1DIR2 akan memiliki logika LOW.
8. Lakukan pengujian untuk arah sebaliknya dengan memberikan perintah **SET MOTOR DIRECTION** motor ke-1 agar berputar berlawanan arah jarum jam. Jika pin-pin pengaturan arah bekerja dengan baik, maka M1DIR1 akan memiliki logika LOW dan M1DIR2 akan memiliki logika HIGH.
9. Lakukan pengujian yang sama untuk pasangan pin-pin pengaturan arah M2DIR1 dan M2DIR2, M3DIR1 dan M3DIR2, serta M4DIR1 dan M4DIR2, masing-masing dengan memberikan perintah untuk mengatur arah motor ke-2, motor ke-3, dan motor ke-4.
10. Lakukan pengujian pin-pin PWM (M1PWM, M2PWM, M3PWM, dan M4PWM) dengan memberikan perintah **SET PWM** untuk masing masing motor. Jika nilai PWM yang diberikan 255 (*duty cycle* 100%), maka pin PWM akan memiliki tegangan sekitar 5 V. Sebaliknya, jika nilai PWM yang diberikan 0 (*duty cycle* 0%), maka pin PWM akan memiliki tegangan 0 V.

6. CONTOH APLIKASI DAN PROGRAM

Pada contoh aplikasi **duaStepper.prj** (disertakan pada CD), SPC MOTOR CONTROLLER digunakan untuk mengendalikan 2 buah motor stepper dengan antarmuka I^2C. Program ditulis dengan menggunakan CodeVisionAVR 1.25.2 versi evaluasi. DT-AVR Low Cost Nano System digunakan sebagai *master* I^2C yang akan mengirimkan perintah. Sebuah modul EMS 2 A DUAL H-BRIDGE digunakan sebagai *driver* motor stepper 1. Sedangkan 2 buah modul EMS 5 A

H-BRIDGE digunakan sebagai *driver* motor stepper 2 (dimisalkan motor stepper 2 memiliki kebutuhan arus yang lebih besar dari 2 A dan lebih kecil dari 5 A).

Berikut koneksi antara modul- modul yang digunakan:



Pada program tersebut, motor stepper 1 diperintahkan untuk bergerak ke posisi 180° dengan menggunakan FullStep dan *step rate* 10 step/detik (stepper 1 dimisalkan memiliki ketelitian 7,5° per step sehingga 180° akan dicapai dalam 24 *step*). Dengan *step rate* sebesar 10 step/detik, maka posisi 180° akan dapat dicapai dalam waktu 2,4 detik.

Sedangkan motor stepper 2 diperintahkan untuk bergerak ke posisi 90° dengan menggunakan microstep32 dan *step rate* 5 step/detik (stepper 2 dimisalkan juga memiliki ketelitian 7,5° per step sehingga 90° akan dicapai dalam 12 *step*). Dengan *step rate* sebesar 5 step/detik, maka posisi 90° akan dapat dicapai dalam waktu 2,4 detik.

Gerakan motor stepper 2 akan terlihat lebih mulus karena dengan menggunakan microstep32, satu *step* motor stepper (misal dalam hal ini 7,5°) akan dibagi menjadi 32 *sub step* (untuk tipe *step* microstep32).

Setelah kedua motor stepper mencapai posisi tujuannya, keduanya diperintahkan untuk kembali ke posisi 0° dengan *step rate* 50 step/detik. Setelah kedua motor stepper kembali ke posisi awalnya, program akan melakukan *looping* untuk kembali melakukan tugas sebelumnya.

- *Terima Kasih atas kepercayaan Anda menggunakan produk kami, bila ada kesulitan, pertanyaan atau saran mengenai produk ini silahkan menghubungi technical support kami :*

support@innovativeelectronics.com

LAMPIRAN
Skematik SPC MOTOR CONTROLLER