
UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
PERPUSTAKAAN

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	**: Ferdi Andika**
NPM	**: 0606031881**
Tanda Tangan	**:**
Tanggal	**: 15 Juni 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	: Ferdi Andika
NPM	: 0606031881
Program Studi	: Teknik Elektro
Judul Skripsi	: Implementasi EMC2 sebagai Pengendali Mesin CNC Berbasis *Open Source*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing :
Ir. Wahidin Wahab, M.Sc, Ph.D ( )

Penguji 1 :
Dr. Abdul Muis, S.T, M.Eng. ( )

Penguji 2 :
Dr. Abdul Halim, M.Eng ( )

Ditetapkan di : Depok

Tanggal : 2 Juli 2010

UCAPAN TERIMA KASIH

Puji syukur kepada Allah SWT atas segala rahmat dan hidayah-Nya sehingga skripsi ini dapat diselesaikan. Shalawat dan salam semoga senantiasa tercurahkan kepada Nabi Muhammad SAW. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, sangat sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada :

1. Bapak Ir. Wahidin Wahab, M. Sc., Ph.D., selaku pembimbing yang telah meluangkan waktunya untuk memberikan bimbingan, saran, dan pengarahan dan kemudahan lainnya sehingga skripsi ini dapat diselesaikan dengan baik;
2. Bapak Dr. Abdul Muis, M. Eng., yang telah meluangkan waktunya untuk memberikan saran dan pengarahan dalam pembuatan skripsi ini;
3. Orang tua dan keluarga saya yang telah memberikan semangat serta dukungan material dan moral;
4. Candraditya Pradhana dan Deni Umaryadi yang telah membantu saya dalam pembuatan skripsi ini;
5. Teman dan sahabat saya yang telah memberikan semangat dan informasi untuk menyelesaikan skripsi ini;
6. Pak Tarki dan Pak Naseh yang telah meluangkan waktunya untuk menunggu saya dalam pengambilan data; dan
7. Seluruh pihak yang telah membantu kami dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, Juni 2010

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Ferdi Andika
NPM	: 0606031881
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (Non-exclusive RoyaltyFree Right) atas karya ilmiah saya yang berjudul :

IMPLEMENTASI EMC2 SEBAGAI PENGENDALI MESIN CNC BERBASIS OPEN SOURCE

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada Tanggal : 15 Juni 2010

Yang menyatakan

(Ferdi Andika)

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	**: Ferdi Andika**
NPM	**: 0606031881**
Tanda Tangan	**:**
Tanggal	**: 15 Juni 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	: Ferdi Andika
NPM	: 0606031881
Program Studi	: Teknik Elektro
Judul Skripsi	: Implementasi EMC2 sebagai Pengendali Mesin CNC Berbasis *Open Source*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing :
Ir. Wahidin Wahab, M.Sc, Ph.D ( )

Penguji 1 :
Dr. Abdul Muis, S.T, M.Eng. ( )

Penguji 2 :
Dr. Abdul Halim, M.Eng ( )

Ditetapkan di : Depok

Tanggal : 2 Juli 2010

UCAPAN TERIMA KASIH

Puji syukur kepada Allah SWT atas segala rahmat dan hidayah-Nya sehingga skripsi ini dapat diselesaikan. Shalawat dan salam semoga senantiasa tercurahkan kepada Nabi Muhammad SAW. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, sangat sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada :

1. Bapak Ir. Wahidin Wahab, M. Sc., Ph.D., selaku pembimbing yang telah meluangkan waktunya untuk memberikan bimbingan, saran, dan pengarahan dan kemudahan lainnya sehingga skripsi ini dapat diselesaikan dengan baik;
2. Bapak Dr. Abdul Muis, M. Eng., yang telah meluangkan waktunya untuk memberikan saran dan pengarahan dalam pembuatan skripsi ini;
3. Orang tua dan keluarga saya yang telah memberikan semangat serta dukungan material dan moral;
4. Candraditya Pradhana dan Deni Umaryadi yang telah membantu saya dalam pembuatan skripsi ini;
5. Teman dan sahabat saya yang telah memberikan semangat dan informasi untuk menyelesaikan skripsi ini;
6. Pak Tarki dan Pak Naseh yang telah meluangkan waktunya untuk menunggu saya dalam pengambilan data; dan
7. Seluruh pihak yang telah membantu kami dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, Juni 2010

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Ferdi Andika
NPM	: 0606031881
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (Non-exclusive RoyaltyFree Right) atas karya ilmiah saya yang berjudul :

IMPLEMENTASI EMC2 SEBAGAI PENGENDALI MESIN CNC BERBASIS OPEN SOURCE

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada Tanggal : 15 Juni 2010

Yang menyatakan

(Ferdi Andika)

ABSTRAK

Nama : Ferdi Andika
Program Studi : Teknik Elektro
Judul : IMPLEMENTASI EMC2 SEBAGAI PENGENDALI MESIN CNC BERBASIS OPEN SOURCE

Persaingan dunia yang ketat menuntut industri untuk meningkatkan kualitas produk dan efisiensi. Untuk itu industri harus tetap mengikuti perkembangan teknologi sistem produksi. Salah satu teknologi tersebut adalah mesin *computer numerical control* (CNC). Permasalahan utama dari mesin CNC adalah harganya yang sangat mahal. Salah satu cara untuk mengurangi harga mesin CNC yaitu dengan menggunakan sistem operasi yang bersifat *open source*. Oleh karena itu, dalam skripsi ini akan dibahas implementasi *Enhanced Machine Control* 2 (EMC2) sebagai *software* pengendali mesin CNC yang berbasis Linux Ubuntu. Pengujian dilakukan pada mesin CNC dengan motor AC-Servo SGMAH-02AAA21, SGDM-02ADA sebagai servopack, dan Mesa 5I20 beserta 7I33 sebagai interface. Analisis yang dilakukan terbatas pada data yang diperoleh dari encoder. Hasil percobaan dan analisis pada skripsi ini menunjukkan nilai parameter pada modul PID yang baik untuk digunakan pada mesin yang dirancang. Parameter tersebut adalah P=100, D=0.005, dan deadband=0.005.

Kata kunci :
mesin CNC, SGMAH-02AAA21, SGDM-02ADA, Mesa 5I20, Mesa 7I33, *open source*, *Enhanced Machine Control 2* (EMC2)

ABSTRACT

Name : Ferdi Andika
Study Program : Electrical Engineering
Title : IMPLEMENTATION OF EMC2 AS OPEN SOURCE CNC MACHINE CONTROLLER

Tight global competition has demanded the industries to keep on improving product's quality and efficiency. Therefore, they have to follow the development of production's system technology. One of these technologies is the computer numerical control (CNC) machine. The main problem of the CNC machine is its extremely high cost. One way to reduce the cost is by using open source operating system. Hence, this thesis will describe the implementation of the Linux-based *Enhanced Machine Control* 2 (EMC2) as a controller software of the machine. The testing was done on the CNC machine with SGMAH-02AAA21 AC-Servo motor, SGDM-02ADA as servopack and Mesa 5I20 plus 7I33 as interface. The analysis of performance in the thesis is limited on the data obtained from the encoder. The result of the experiment and the analysis show good PID parameters' value which can be implemented in the designed machine. Those parameters are P=100, D=0.005, and deadband=0.005.

Key words :
CNC machine, SGMAH-02AAA21, SGDM-02ADA, Mesa 5I20, Mesa 7I33, open source, Enhanced Machine Control 2 (EMC2)

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR ISTILAH

duty cycle: perbandingan lama pulsa high dengan satu periode sinyal PWM atau PDM

feederate : kecepatan motor yang menggambarkan seberapa jauh motor dapat memotong dalam satuan waktu. Satuannya inci/menit atau mm/menit

G-Code : bahasa standar dalam penulisan instruksi untuk menjalankan mesin CNC

set point : sinyal referensi pada suatu pengendali

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Persaingan dunia yang ketat menuntut industri untuk meningkatkan efisiensi, kualitas produk, dan parameter lainnya. Persaingan tersebut memaksa industri untuk tetap mengikuti perkembangan teknologi sistem produksi. Salah satu teknologi yang banyak dipakai industri adalah mesin *computer numerical control* (CNC).

Permasalahan utama dari mesin CNC adalah harganya yang mahal. Hal ini menyebabkan tidak semua industri bisa memanfaatkan teknologi ini. Industri kecil dan menengah tidak mampu membeli mesin-mesin produksi yang mahal ini sehingga kualitas produknya rendah dan harga produknya tinggi, akibatnya industri kecil dan menengah sulit bersaing.

Salah satu cara untuk menurunkan harga mesin CNC yaitu dengan memakai sistem operasi berbasis *open source*. Dengan cara itu, harga dapat ditekan dari segi sistem operasi dan *software* yang digunakan untuk mengendalikan mesin CNC. Oleh karena itu, dalam skripsi ini akan dibahas implementasi EMC2 berbasis Ubuntu Linux untuk mengendalikan mesin CNC.

1.2 Tujuan

Tujuan dari skripsi ini adalah :

1. Membahas prinsip kerja EMC2 berbasis open source.
2. Membahas *file* konfigurasi pada EMC2 untuk mengendalikan mesin CNC yang menggunakan motor AC-servo sebagai penggerak.
3. Mengimplementasikan EMC2 sebagai pengendali mesin CNC berbasis open source

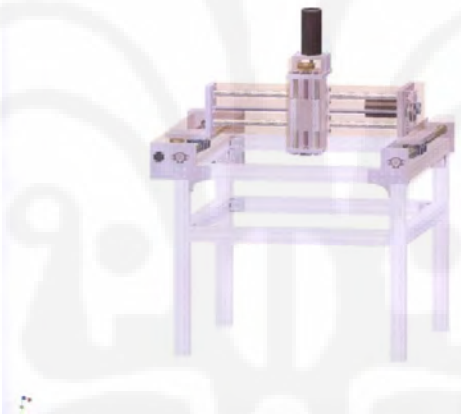
1.3 Batasan Masalah

Pembahasan pada skripsi ini ditekankan pada pengonfigurasian EMC2 untuk mengoptimalkan kinerja mesin CNC. Kemudian akan dilakukan pengujian untuk mengetahui performansi mesin CNC. Mesin CNC tidak dibuat dalam

skripsi ini, tetapi merupakan perangkat yang dirancang dalam salah satu penelitian di lab teknik kendali FT UI.

1.4 Sistematika Penulisan

Penulisan buku skripsi ini dibagi menjadi 5 bab. Bab 1 merupakan bab pendahuluan yang memuat latar belakang, tujuan, batasan masalah, dan sistematika penulisan dari skripsi ini. Bab 2 berisi tinjauan literatur tentang konsep alat-alat yang diterapkan pada EMC2, prinsip kerja EMC2, dan teori lain yang mendukung dalam implementasi EMC2 untuk mesin CNC. Bab 3 berisi tentang penjelasan konfigurasi EMC2 pada mesin CNC yang dirancang. Bab 4 berisi analisa dari performansi mesin CNC akibat konfigurasi EMC2. Bab 5 berisi kesimpulan dari hasil analisa pengujian performansi mesin CNC.



- M : Miscellaneous function

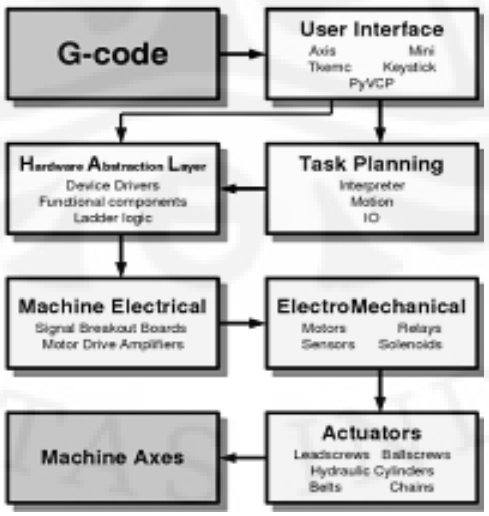
Contoh : G01 X1.0 F15. Intruksi ini memerintahkan motor pada x axis untuk bergerak lurus ke koordinat X=1.0 dengan feedrate 15 inci/menit.

Pada sistem produksi modern, *end-to-end* desain untuk komponen yang akan diproduksi, dilakukan menggunakan CAD/CAM sistem, untuk meningkatkan tingkat otomatisasi. Program CAD/CAM mengubah sebuah *file* komputer menjadi instruksi G-Code yang dibutuhkan untuk menjalankan mesin CNC.

2.2 Enhanced Machine Control 2 (EMC2)

Enhanced Machine Control 2 (EMC2) merupakan sebuah *software* berbasis Ubuntu Linux yang berfungsi sebagai interpreter G-Code dan sekaligus sebagai pengendali mesin. EMC2 merupakan pengembangan dari *software* sebelumnya yaitu EMC. EMC2 banyak digunakan untuk pengendalian mesin CNC tetapi *software* ini bisa diimplementasikan pada mesin lain seperti robot dan peralatan otomatis lainnya. *Software* ini bisa mengendalikan motor, relay, dan mesin-mesin lain yang sejenis.

Ada 4 komponen utama pada EMC2 yaitu : pengendali gerak, pengendali I/O diskrit, *task executor*, dan *graphical user interface* (GUI). Sebagai tambahan ada sebuah layer yang disebut HAL (*Hardware Abstraction Layer*) agar EMC2 dapat dikonfigurasi tanpa harus mengompilasi EMC2 kembali. Hubungan antara EMC2, HAL, dan perangkat keras dapat dilihat pada Gambar 2.2.



Gambar 2.2. Diagram alir mesin dengan EMC2

2.2.1 Prinsip Pengendalian pada EMC2

Ada dua hal penting yang menjadi acuan dari performansi mesin CNC yaitu : ketelitian jarak dan ketelitian kecepatan. Mesin CNC harus bisa digerakkan ke posisi tertentu dengan kecepatan yang ditentukan. Hal ini menyebabkan mesin CNC harus memiliki sistem kendali posisi dan sistem kendali kecepatan.

Pada penggunaannya, EMC2 hanya bisa diatur untuk satu jenis pengendali untuk masing-masing axis dengan menggunakan PID. Oleh karena itu, EMC2 membuat suatu metode yang memungkinkan mesin dapat dikendalikan secara posisi dan kecepatan sekaligus dengan trajectory control.

Pada saat user memasukkan perintah FXXX, EMC2 menghitung kecepatan dan akselerasi. Untuk mencapai suatu tujuan, EMC2 memperbarui referensi posisi setiap periode servo. Besar penambahan posisi tersebut disesuaikan dengan besar kecepatan dan akselerasinya. Sehingga seolah-olah, dapat dilakukan pengendalian kecepatan dengan menggunakan pengendalian posisi.

2.2.2 Konfigurasi EMC2

Untuk mengendalikan sebuah mesin, dibutuhkan 5 (lima) jenis *file* konfigurasi. File konfigurasi ini disesuaikan dengan karakteristik mesin dan perilaku mesin yang diinginkan. File-file tersebut adalah :

1. File .ini, berisi spesifikasi dan informasi penting lain tentang mesin
2. File .hal, berisi konfigurasi HAL (perilaku mesin)
3. File .var, berfungsi untuk menyimpan nilai yang digunakan oleh interpreter
4. File .tbl, file yang menyimpan informasi tentang tool yang digunakan
5. File .nml, berisi konfiigurasi jalur komunikasi antara PC dan mesin

2.2.3 Port Interface

EMC2 dapat dihubungkan dengan mesin menggunakan beberapa port interface pada komputer. Semua pengaturan penggunaan port ini dilakukan pada pengonfigurasian file .hal. Secara default, EMC2 hanya bisa dihubungkan dengan mesin menggunakan port serial dan port parallel. Port interface lain hanya bisa digunakan dengan terlebih dahulu membuat firmware sendiri untuk melakukan

komunikasi dengan EMC2, seperti Mesa 5i20 dengan HostMot2 sebagai firmwarenya. Tabel 2.1 menunjukkan karakteristik dari beberapa port interface saat digunakan pada EMC2.

Tabel 2.1. Karakteristik port interface EMC2

No	Karakteristik	Parallel Port	Serial Port	Mesa 5i20
1	Pin I/O	17	7	72
2	Konfigurasi switch	Membutuhkan pull-up resistor	Membutuhkan pull-up resistor	Tidak membutuhkan pull-up resistor
3	Signal generator	PC	PC	FPGA
4	Pengolahan sinyal dari ecoder	PC	PC	FPGA
5	Tegangan output	0V, 5V	Tegangan RS232	0V, 3.3 V

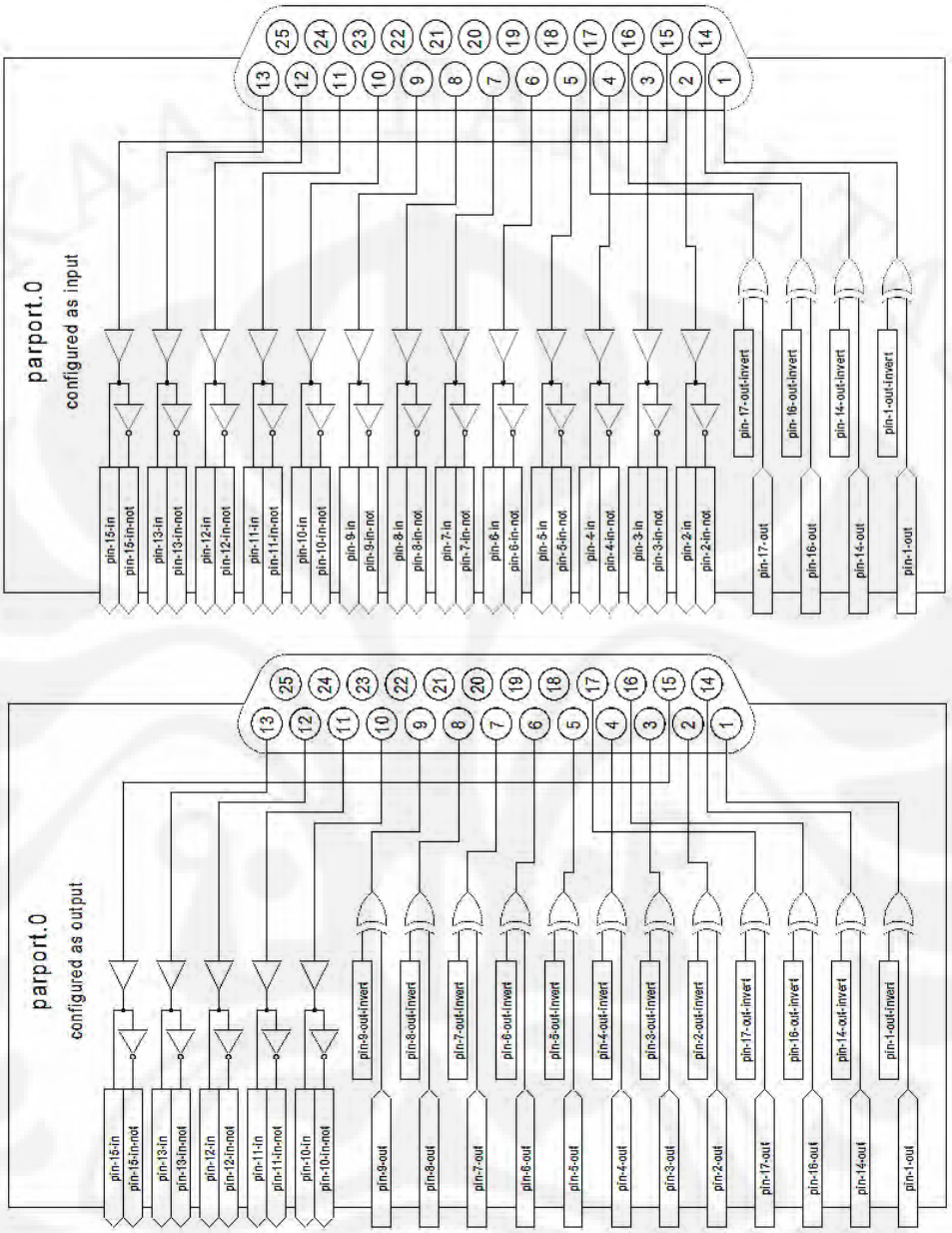
Untuk komunikasi antara komputer dan mesin dengan port parallel, digunakan komponen `hal_parport` pada HAL. Port ini mempunyai 17 I/O yang bisa digunakan. Pin 1, 14, 16, dan 17 bertindak sebagai ouput sedangkan pin 10, 11, 12, 13, dan 15 bertindak sebagai input. Pin 2 sampai 9 bisa di set sebagai input atau output, jika diset sebagai input maka semua pin 2 sampai 9 akan bertindak sebagai input dan jika diset sebagai output maka semua pin 2 sampai 9 akan bertindak sebagai output (Gambar 2.3).

Dalam pengonfigurasiannya, parallel port mempunyai 3 jenis pin, yaitu :

1. `(BIT) parport.<portnum>.pin-<pinnum>-out`
2. `(BIT) parport.<portnum>.pin-<pinnum>-in`
3. `(BIT) parport.<portnum>.pin-<pinnum>-in-not`

Parameter pada parallel port yang bisa digunakan adalah :

1. `(BIT) parport.<portnum>.pin-<pinnum>-out-invert`
2. `(BIT) parport.<portnum>.pin-<pinnum>-out-reset` (hanya untuk pin 2-9)
3. `(U32) parport.<portnum>.reset-time` (dalam nanoseconds)


parport.0
configured as input
pin-15-in
pin-15-in-not
pin-13-in
pin-13-in-not
pin-12-in
pin-12-in-not
pin-11-in
pin-11-in-not
pin-10-in
pin-10-in-not
pin-9-in
pin-9-in-not
pin-8-in
pin-8-in-not
pin-7-in
pin-7-in-not
pin-6-in
pin-6-in-not
pin-5-in
pin-5-in-not
pin-4-in
pin-4-in-not
pin-3-in
pin-3-in-not
pin-2-in
pin-2-in-not
pin-17-out-invert
pin-16-out-invert
pin-14-out-invert
pin-1-out-invert
pin-17-out
pin-16-out
pin-14-out
pin-1-out
parport.0
configured as output
pin-15-in
pin-15-in-not
pin-13-in
pin-13-in-not
pin-12-in
pin-12-in-not
pin-11-in
pin-11-in-not
pin-10-in
pin-10-in-not
pin-9-out-invert
pin-8-out-invert
pin-7-out-invert
pin-6-out-invert
pin-5-out-invert
pin-4-out-invert
pin-3-out-invert
pin-2-out-invert
pin-17-out-invert
pin-16-out-invert
pin-14-out-invert
pin-1-out-invert
pin-9-out
pin-8-out
pin-7-out
pin-6-out
pin-5-out
pin-4-out
pin-3-out
pin-2-out
pin-17-out
pin-16-out
pin-14-out
pin-1-out

3. `serport.N.pin-8-in`, pin 5 pada pinout port serial 25-pin
4. `serport.N.pin-9-in`, pin 22 pada pinout port serial 25-pin
5. `serport.N.pin-1-in-not`, versi invert dari pin-1-in
6. `serport.N.pin-6-in-not`, versi invert dari pin-6-in
7. `serport.N.pin-8-in-not`, versi invert dari pin-8-in
8. `serport.N.pin-9-in-not`, versi invert dari pin-9-in
9. `serport.N.pin-3-out`, pin 2 pada pinout port serial 25-pin
10. `serport.N.pin-4-out`, pin 20 pada pinout port serial 25-pin
11. `serport.N.pin-7-out`, pin 4 pada pinout port serial 25-pin

Parameter yang dapat digunakan pada port serial adalah :

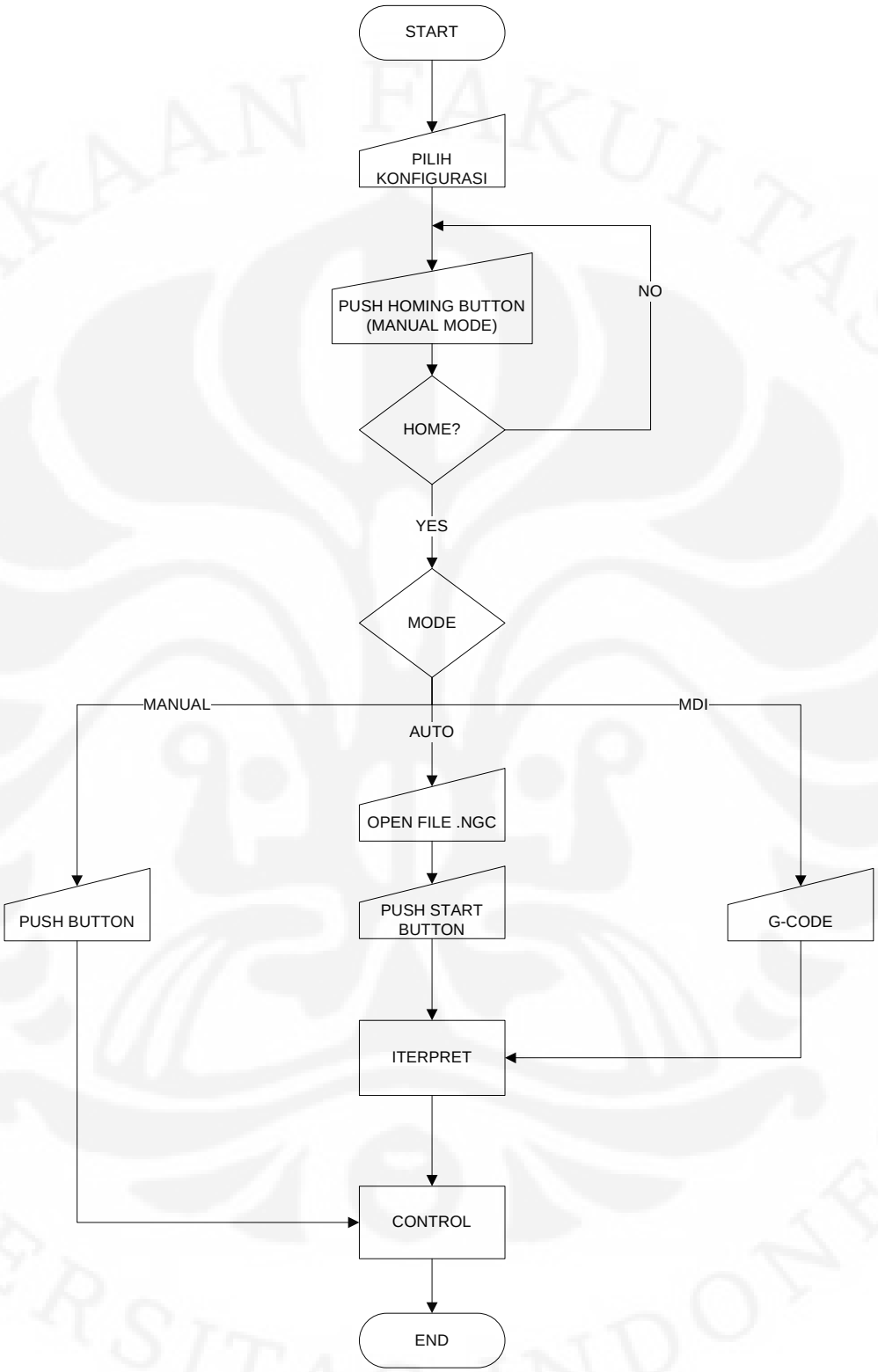
1. `serport.N.pin-3-out-invert`
2. `serport.N.pin-4-out-invert`
3. `serport.N.pin-7-out-invert`
4. `serport.N.ioaddr` (alamat port serial yang digunakan)

2.2.4 Mode Operasi

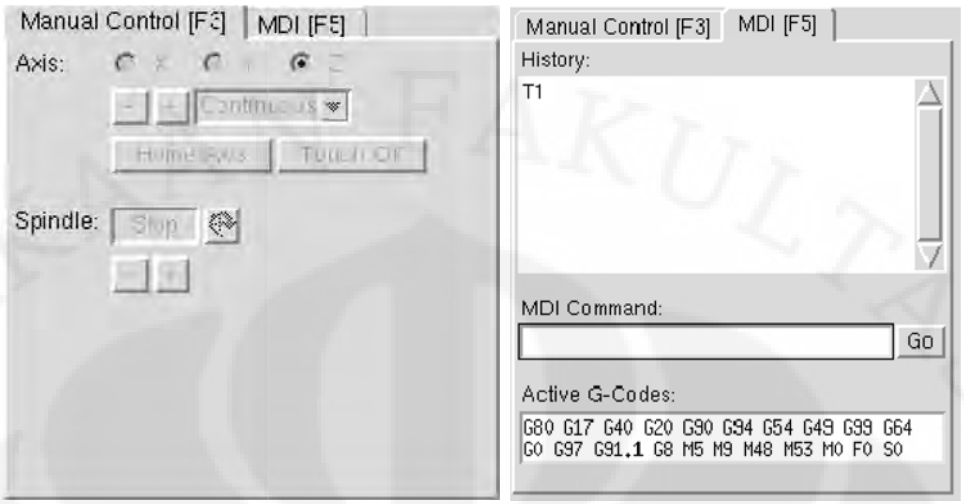
Setelah pengonfigurasian EMC2 dilakukan dengan benar maka mesin CNC telah siap untuk dijalankan. Gambar 2.4 merupakan diagram alir saat menjalankan EMC2.

Ada 3 mode untuk menjalankan mesin dengan EMC2. Mode tersebut adalah mode manual, auto, dan MDI. Semua mode ini disatukan dalam suatu GUI (Gambar 2.5). Ada beberapa hal yang hanya bisa dilakukan pada mode tertentu. Operator bisa memerintahkan axis untuk bergerak ke posisi home hanya pada mode manual. Operator bisa menjalankan mesin dengan mengeksekusi file yang berisi G-Code hanya pada mode auto.

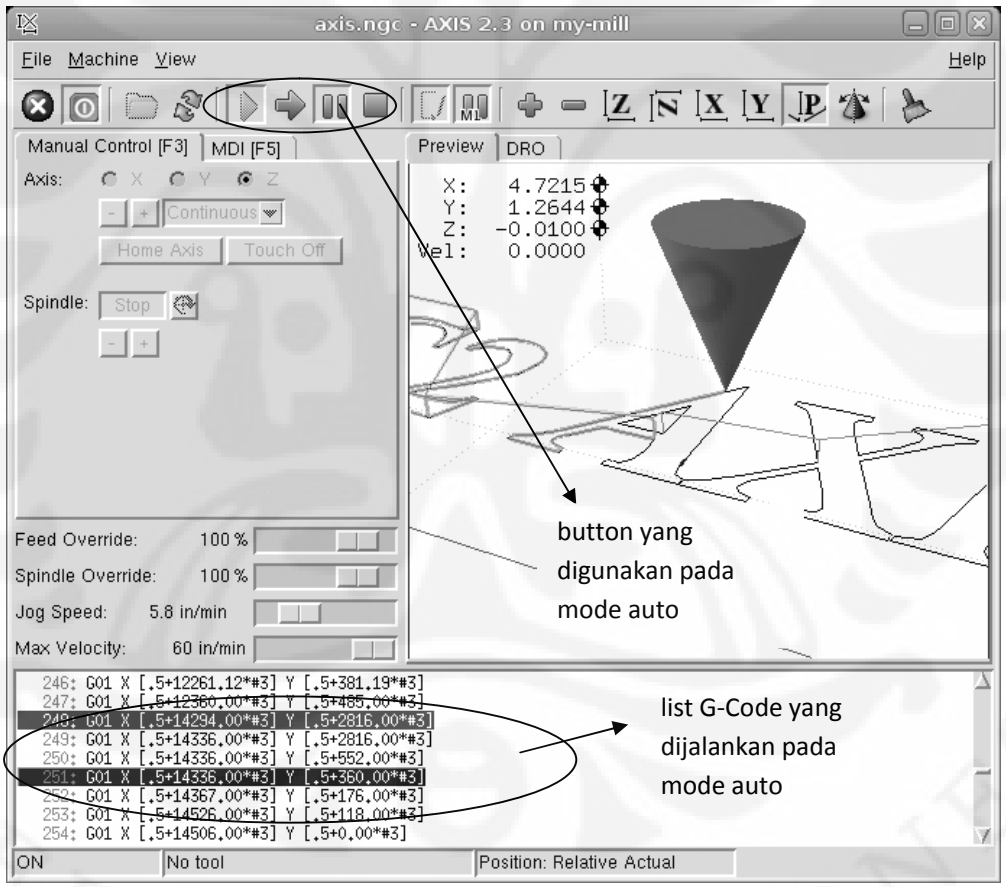
Pada mode manual, tiap perintah dimasukkan secara terpisah, Misalnya menghidupkan pendingin, *jog* X-axis dengan kecepatan 25 inci per menit. Tiap perintah dilakukan dengan menekan tombol pada user interface atau menekan tombol pada keyboard. Pada mode auto, user menjalankan mesin dengan membuka file .ngc yang berisi urutan G-Code yang akan dijalankan. Pada mode MDI, user memasukkan perintah dengan mengetikkan G-Code secara manual dan menekan <return> atau tombol <enter> pada keyboard.



Gambar 2.4. Diagram alir menjalankan EMC2



(a) (b)



(c)

Gambar 2.5. AXIS interface, (a)manual, (b)MDI, (c)auto

2.3 Firmware HostMot2

EMC2 mendukung *firmware* Hostmot2 yang dibuat oleh Mesa dengan adanya *generic driver* yang disebut dengan hostmot2 dan 2 *low-level I/O driver* untuk *anything I/O board*. *Low-level I/O driver* tersebut adalah hm2_7i43 dan hm2_pci. Kedua *low-level I/O driver* ini berfungsi untuk membuat komunikasi antara FPGA dengan komputer melalui antar muka yang digunakan pada Mesa *card.*

Ada beberapa modul yang terdapat pada *firmware* HostMot2. Setiap modul mempunyai fungsi, parameter, dan pin tertentu. Modul-modul tersebut yaitu : watchdog, GPIO, stepgen, PWMgen, dan encoder.

2.3.1 Modul Watchdog

Modul ini berfungsi sebagai pemutus semua komunikasi dengan Mesa *board* dan melakukan *reset* pin I/O. Agar komunikasi tidak terputus, *function* `.pet-watchdog` harus dipanggil sebelum waktu *timeout_ns*. Pin yang ada pada modul ini adalah :

- `.has_bit` – jika bernilai *true*, user bisa mengembalikan operasi dengan mengubah nilai *pin* menjadi *false*

Parameter yang ada pada modul ini adalah :

- `.timeout_ns (u32, RW)` – *timeout* pada *watchdog*. Nilai *default* dari *parameter* ini adalah 1.000.000.000 (1 detik)

2.3.2 Modul GPIO

Modul ini berfungsi sebagai digital I/O untuk keperluan tertentu sesuai rancangan HAL yang dibuat. Pin yang ada pada modul ini adalah :

- `.in (Bit, Out)` – merupakan input yang masuk melalui *physical_pin*
- `.in_not (Bit, Out)` – merupakan nilai *invert* dari input
- `.out (Bit, In)` – merupakan output ke *physical_pin*

Parameter yang ada pada modul ini adalah :

- `.invert_output (Bit, RW)` – *parameter* ini hanya berfungsi jika *parameter* `.is_output` bernilai *true*. Jika *parameter* ini bernilai *true*, maka nilai output yang dikeluarkan *pin* `.out` akan di-*invert*

- `.is_opendrain (Bit, RW)` – *parameter* ini hanya berfungsi jika *parameter* `.is_output` bernilai *true*. Jika *parameter* ini bernilai *true*, maka impedansi pada *pin* akan tinggi
- `.is_output (Bit, RW)` – jika *parameter* ini bernilai *false*, maka *pin* GPIO yang berfungsi hanya *pin* input. Jika *parameter* ini bernilai *true*, maka GPIO berfungsi sebagai output

2.3.3 Modul PWMgen

Modul PWMgen merupakan modul yang berfungsi untuk menghasilkan sinyal PWM. Pin yang ada pada modul ini adalah :

- `.enable (Bit, In)` – *pin* untuk mengaktifkan modul PWMgen
- `.value (Float, In)` – referensi yang diinginkan

Parameter yang ada pada modul ini adalah :

- `.output-type (s32, RW)` – merupakan tipe dari output yang diinginkan. Nilai yang diterima pada *parameter* ini adalah :
  1 – PWM pada Out0 dan Dir pada Out1
  2 – Up on Out0 dan Down pada Out1
  3 – mode PDM, PDM pada Out0 dan Dir pada Out1
  4 – Dir pada Out0 dan PWM pada Out1
- `.scale (Float, RW)` – faktor skala untuk mengubah unit posisi ke *duty cycle* (dc = posisi/skala). *Duty cycle* mempunyai kisaran efektif yaitu dari -1.0 sampai dengan 1.0, diluar dari itu akan dipotong
- `.pdm_frequency (u32, RW)` – merupakan frekuensi PDM yang ada pada tipe 3 dari modul PWMgen
- `.pwm_frequency (u32, RW)` – merupakan frekuensi PWM yang ada pada tipe 1 dan 2 dari modul PWMgen

2.3.4 Modul Encoder

Modul encoder merupakan modul yang berfungsi untuk melakukan perhitungan pada sinyal masukan dari encoder (*hardware*). Pin yang ada pada modul ini adalah :

- `.count (s32, Out)` – jumlah *count* dari encoder setelah direset
- `.index-enable (Bit, I/O)` – saat *pin* ini diset *true*, *count* akan direset saat ada *index pulse*.
- `.position (Float, Out)` – posisi *encoder* (*count/scale*)
- `.rawcounts (s32, Out)` – total *count* dari awal mesin *start*
- `.reset (Bit, In)` – saat *pin* ini bernilai *true*, *count* dan position akan diset 0. *Pin* ini tidak otomatis berubah nilai saat mereset *count* jadi harus diubah secara manual oleh user
- `.velocity (Float, Out)` – estimasi kecepatan dalam unit posisi per detik

Parameter yang ada pada modul ini adalah :

- `.counter-mode (Bit, RW)` – secara *default* akan diset *false* untuk mode quadrature. *Parameter* diset *true* untuk mode *Up/Down* atau mode satu input (*phase* A)
- `.filter (Bit, RW)` – secara *default* akan diset *true*, *quadrature counter* membutuhkan 15 *clock* untuk menyimpan perubahan dari 3 sinyal input (pulsa yang datang lebih cepat akan diabaikan). Jika diset *false*, *quadrature counter* hanya membutuhkan 3 *clock* untuk menyimpan perubahan. *Clock* dari *encoder sample* harus 33 MHz pada PCI AnyIO dan 50 MHz pada 7i43
- `.index-invert (Bit, RW)` – jika diset *true*, *rising edge* dari input *index* akan menyebabkan *index event*. Jika diset *false*, maka *falling edge* dari input *index* yang menyebabkan *index event*
- `.index-mask (Bit, RW)` – jika diset *true*, *physical_pin* dari *index* akan berpengaruh jika *physical_pin* dari *index-mask* bernilai *true*
- `.index-mask-invert (Bit, RW)` – jika diset *true*, *physical_pin* dari *index-mask* harus bernilai *false* agar *index* mempunyai efek
- `.scale (Float, RW)` – faktor skala untuk mengubah dari unit "*count*" menjadi unit posisi
- `.vel-timeout (Float, RW)` – merupakan waktu untuk menunggu pulsa berikutnya sebelum dilaporkan bahwa *encoder* telah berhenti. Satuan *parameter* ini adalah detik.

2.4 Pulse Density Modulation

Pulse density modulation (PDM) merupakan salah satu bentuk modulasi yang berfungsi untuk menyatakan sinyal analog ke dalam bentuk sinyal digital. Pada sinyal PDM, besar amplitudo dinyatakan dalam bentuk kerapatan (densitas). Pengodean sinyal analog pada PDM melalui proses delta-sigma modulation (Gambar 2.5). Algoritma umum PDM adalah sebagai berikut :

```
e[-1] = 0
for n from 0 to sampling
     if x[n] > e[n-1] then y[n]=1
     else y[n]= -1
     e[n] = y[n]-x[n]+e[n-1]
end for
```

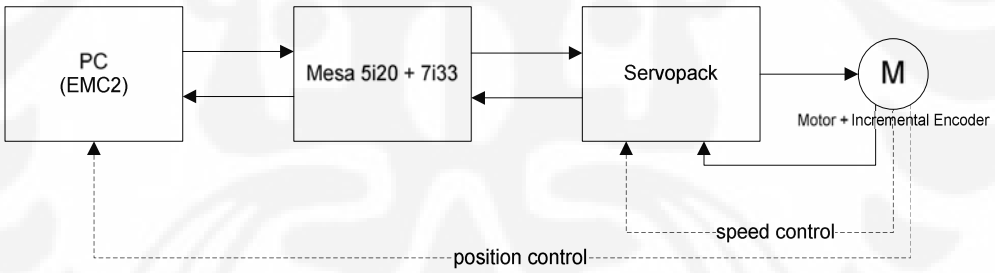
Untuk menggunakan sinyal PDM sebagai set point, sinyal ini harus diubah menjadi sinyal analog. Untuk mengubah sinyal PDM menjadi analog digunakan low-pass filter. Hal ini dikarenakan low-pass filter digunakan untuk merata-ratakan sinyal. Rata-rata amplitudo dari pulsa akan dihitung dengan kerapatan pulsa.

PDM pada Mesa 5I20 dibuat bukan berdasarkan algoritma umum yang telah dijelaskan sebelumnya. PDM pada Mesa 5I20 dibuat berdasarkan duty cycle. Duty cycle ini merupakan representasi perbandingan lama pulsa high dengan satu periode sinyal. Dengan adanya parameter scale pada modul PWMgen HostMot2, nilai input langsung dibagi dengan nilai scale sehingga didapatkan duty cycle dari sinyal. Algoritma pembuatan PDM ini pada Mesa 5I20 ada pada file pwmgen.c pada source code driver mesa-hostmot2. Untuk mengeluarkan output pada modul PWMgen dalam bentuk sinyal PDM, parameter `.output-type` pada modul PWMgen harus diset untuk bernilai 3.

BAB 3
RANCANGAN KONFIGURASI EMC2 DAN RENCANA UJI COBA

3.1 Gambaran Umum Rancangan

Mesin CNC yang dirancang merupakan mesin 3 sumbu dengan menggunakan 3 motor SGMAH-02AAA21, 3 SGDM-02ADA sebagai servopack, 1 PC sebagai pengendali, dan unit Mesa 5i20 beserta unit Mesa 7i33 sebagai *interface* antara PC dan *servopack* (Gambar 3.1). Pada setiap motor terdapat *incremental encoder* sebagai *feedback* yang akan digunakan pada pengendali. Mesin ini menggunakan pengendali cascade, pengendali pertama merupakan pengendali kecepatan yang terdapat pada servopack dan pengendali kedua merupakan pengendali posisi yang terdapat pada EMC2. Unit Mesa 5i20 berfungsi sebagai pembangkit sinyal PDM dan mengolah sinyal *encoder* yang masuk sehingga data posisi langsung dapat digunakan oleh komputer. Unit Mesa 7i33 berfungsi untuk mengubah sinyal PDM dari unit Mesa 5i20 menjadi sinyal analog dengan range +10V sampai degan -10V sebagai referensi kecepatan dari *servopack*.



Gambar 3.1. Blok diagram mesin CNC

Untuk menjalankan mesin dengan EMC2, ada beberapa penyesuaian yang perlu dilakukan karena setiap mesin mempunyai perilaku dan spesifikasi yang berbeda. Penyesuaian ini dilakukan dengan mengubah *file-file* konfigurasi pada EMC2.

3.2 Perancangan File .ini

File .ini berisi tentang data-data yang menjadi konfigurasi mesin. Untuk memudahkan pengonfigurasian file .ini, file ini terbagi menjadi beberapa bagian. Bagian ini dibagi berdasarkan fungsinya.

3.2.1 HOSTMOT2

Mesin CNC yang dirancang menggunakan Mesa 5i20 sebagai interface. Bagian HOSTMOT2 ini berisi tentang konfigurasi firmware untuk Mesa 5i20. Firmware yang digunakan disesuaikan dengan kebutuhan. Mesin CNC yang dirancang menggunakan 3 servo dan 3 encoder. Oleh karena itu, firmware yang digunakan adalah SVST8_4.bit karena firmware ini menyediakan 8 channel encoder, 8 channel PWMgen, dan 4 Stepgen.

3.2.2 EMCMOT

Bagian ini adalah pengaturan pewaktuan yang digunakan dalam perhitungan gerakan servo. Nilai ini didapat dari latency-test yang ada pada EMC2.

3.2.3 TRAJ

AXES merupakan jumlah komponen axis pada HAL. Mesin CNC yang dirancang menggunakan kinematik trivial sehingga jumlah axis dan komponen axis hal bernilai sama.

Untuk mengatur kecepatan maksimum, harus diperhitungkan kecepatan maksimum motor dan kalibrasi dengan mekanik. Kecepatan maksimum 3 motor yang digunakan adalah 3000 rpm atau 50 rps. Kemudian kalibrasi dengan axis yang membutuhkan putaran paling banyak untuk setiap incinya menggunakan feedback encoder. Setelah didapat banyak putaran tiap incinya, maka kecepatan maksimumnya adalah 50 dibagi banyak putaran. Nilai MAX_VELOCITY pada file .ini bisa diset di bawah atau sama dengan kecepatan maksimum axis.

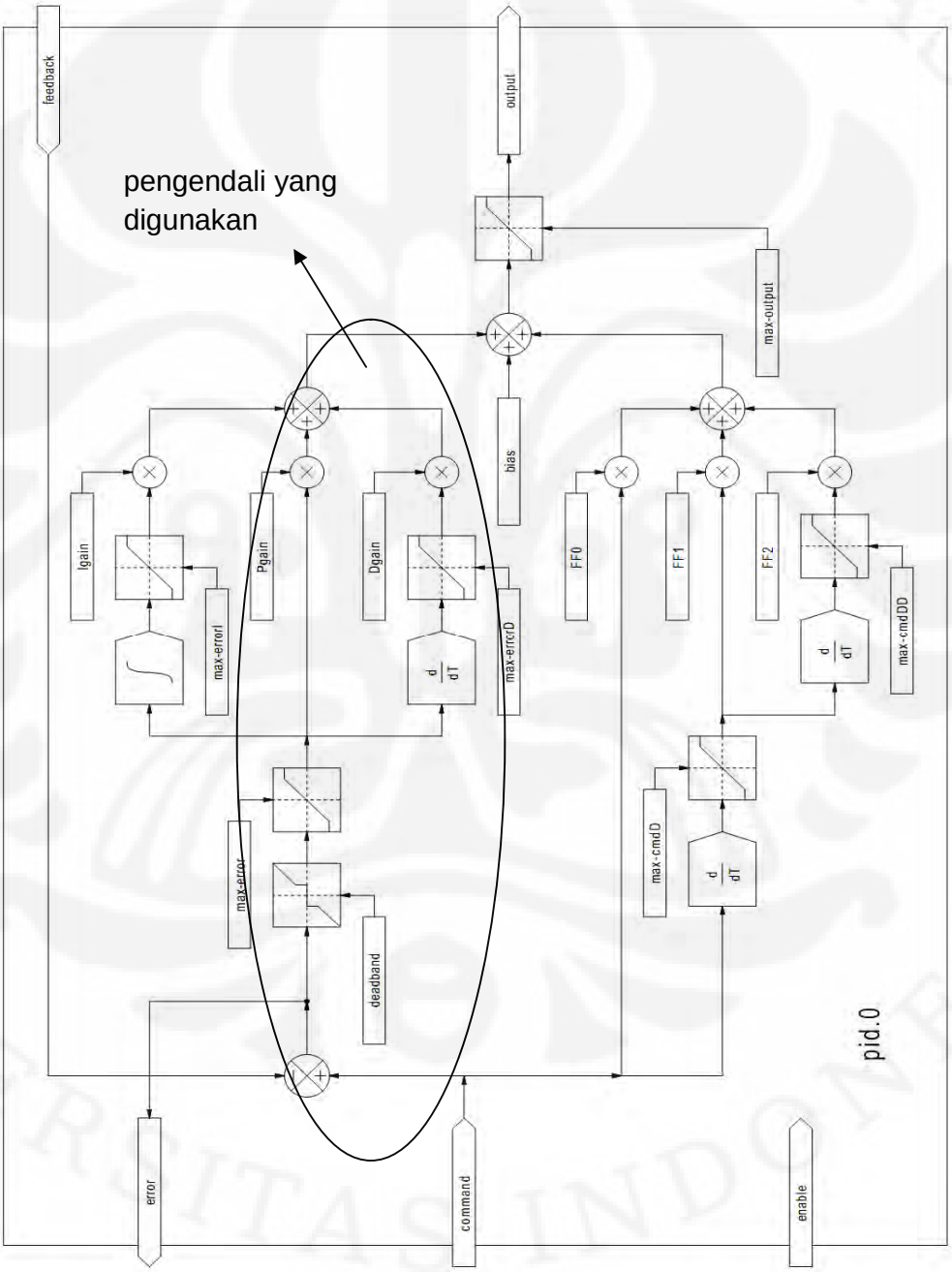
Parameter NO_FORCE_HOMING diset true jika ingin menjalankan mesin tanpa melakukan homing terlebih dahulu. Parameter ini sangat berguna pada saat pengujian satu motor karena lebih fleksibel.

3.2.4 AXIS_n

Parameter INPUT_SCALE dan OUTPUT_SCALE digunakan untuk kalibrasi encoder dengan mekanik. Parameter INPUT_SCALE jumlah count

encoder untuk satu inci. Parameter OUTPUT_SCALE digunakan untuk skala ouput dan dapat digunakan untuk mengubah arah putaran motor.

Pada bagian AXIS_n terdapat parameter pengendali yang akan digunakan pada modul PID pada HAL. Parameter ini ditentukan dengan tuning secara manual. Parameter yang digunakan hanya parameter P, D, dan deadband (Gambar 3.2). Parameter lainnya diset dengan nilai 0.



Gambar 3.2. Blok diagram modul PID pada HAL

3.3 Perancangan File .hal

EMC2 menggunakan HAL sebagai penghubung antara software dan hardware. Hal ini bertujuan agar bisa menyesuaikan dengan banyak tipe mesin tanpa mengompilasi ulang program EMC2. Oleh karena itu, design dari file .hal ini harus disesuaikan dengan mesin yang akan dibuat.

Berikut adalah penjelasan dari isi file .hal yang dirancang :

- **`loadrt trivkins`**

  Kinematik yang digunakan adalah kinemaik trivial. Artinya satu axis digerakkan oleh satu motor.

- **`newsig emcmot.00.enable bit`**
  `sets emcmot.00.enable FALSE`
  `net emcmot.00.enable => pid.0.enable`
  `net emcmot.00.enable =>`
  `hm2_[HOSTMOT2](BOARD).0.pwmgen.00.enable`
  `net emcmot.00.enable <= axis.0.amp-enable-out`

  `axis.0.amp-enable-out` akan aktif jika EMC2 diberi perintah. Oleh karena itu, pin ini dihubungkan dengan `pid.0.enable` dan `pwmgen.00.enable`. Hal ini bertujuan untuk menghindari noise saat EMC2 tidak diberi perintah.

- **`net emcmot.00.pos-cmd axis.0.motor-pos-cmd =>`**
  `pid.0.command`
  `net motor.00.command  pid.0.output  =>`
  `hm2_[HOSTMOT2](BOARD).0.pwmgen.00.value`
  `setp  hm2_[HOSTMOT2](BOARD).0.encoder.00.scale`
  `[AXIS_0]INPUT_SCALE`
  `net motor.00.pos-fb`
  `hm2_[HOSTMOT2](BOARD).0.encoder.00.position =>`
  `pid.0.feedback`
  `net motor.00.pos-fb => axis.0.motor-pos-fb`

  Bagian ini menjelaskan bahwa EMC2 diset menggunakan pengendali posisi. Posisi yang menjadi feedback menggunakan satuan inci seperti yang telah

diset pada bagian file .ini. Oleh karena itu, encoder harus diskalakan dengan factor skala yang terdapat pada file .ini.

- **setp hm2_[HOSTMOT2](BOARD).0.pwmgen.00.output-type 3**
  setp hm2_[HOSTMOT2](BOARD).0.pwmgen.pdm_frequency 6000000
  setp hm2_[HOSTMOT2](BOARD).0.pwmgen.00.scale [AXIS_0]OUTPUT_SCALE

Bagian ini menjelaskan sinyal keluaran dari Mesa 5i20. Hasil keluaran PID posisi merupakan referensi (set point) dari pengendali PI pada servopack. Oleh karena itu, dibutuhkan sinyal analog dengan tegangan yang dapat diubah-ubah. Untuk itu, digunakan sinyal dengan tipe PDM dengan frekuensi 6Mhz seperti yang telah dijelaskan pada Bab 2.

BAB 4
ANALISIS DATA

Setelah perancangan file konfigurasi EMC2, perlu adanya pengujian terhadap mesin CNC untuk membuktikan performansinya. Bab analisis data akan membahas pengujian pada mesin CNC dan akan memaparkan analisis yang berkaitan dengan performansi mesin.

4.1 Pengujian Satu Motor

Pengujian dilakukan terhadap satu motor dengan beberapa langkah, yaitu :

1. Menjalankan motor dengan memasukkan G-Code tertentu pada EMC2 dengan posisi awal pada titik koordinat cartesian (0,0,0).
2. Membaca pin-pin HAL dengan halscope EMC2. Pin-pin tersebut adalah :
   - Pin pid.0.command, referensi pada modul PID EMC2
   - Pin encoder.00.position, feedback posisi dari encoder
   - Pin pid.0.output, output pada modul PID EMC2
   - Pin encoder.00.velocity, diferensiasi feedback posisi
3. Kemudian hasil pembacaan pin-pin HAL disimpan dalam file ‘’.log.

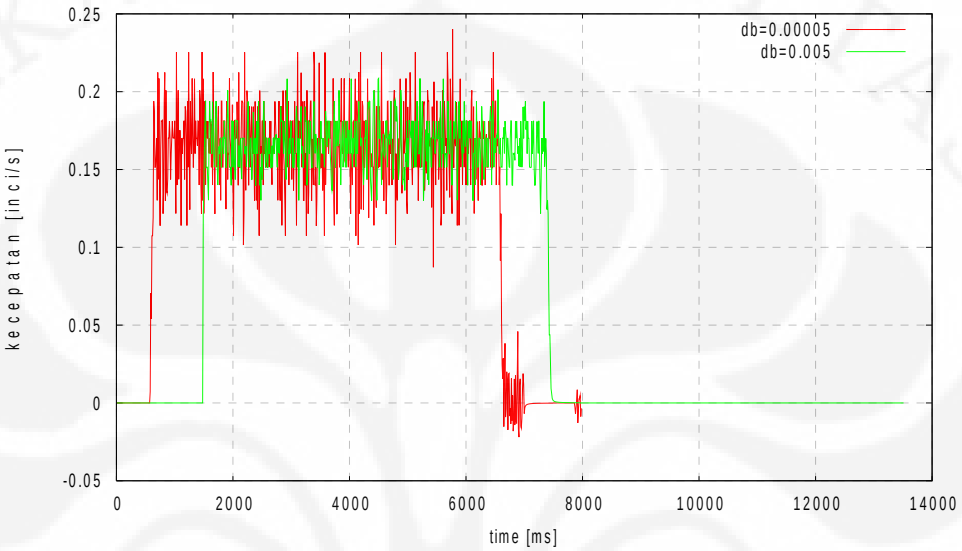
Selanjutnya akan dipaparkan hasil uji coba dan analisis dari data yang diperoleh.

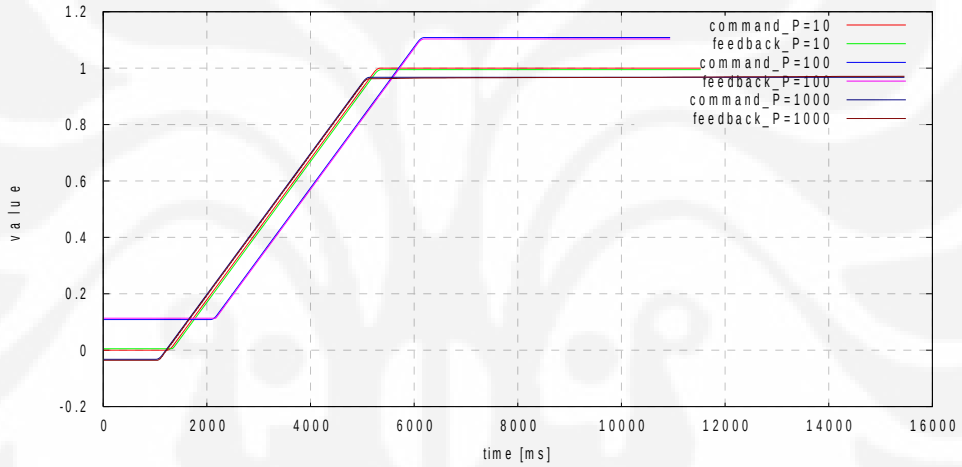
4.1.1 Variasi Parameter Deadband pada Modul PID EMC2

Percobaan dilakukan dengan kriteria :

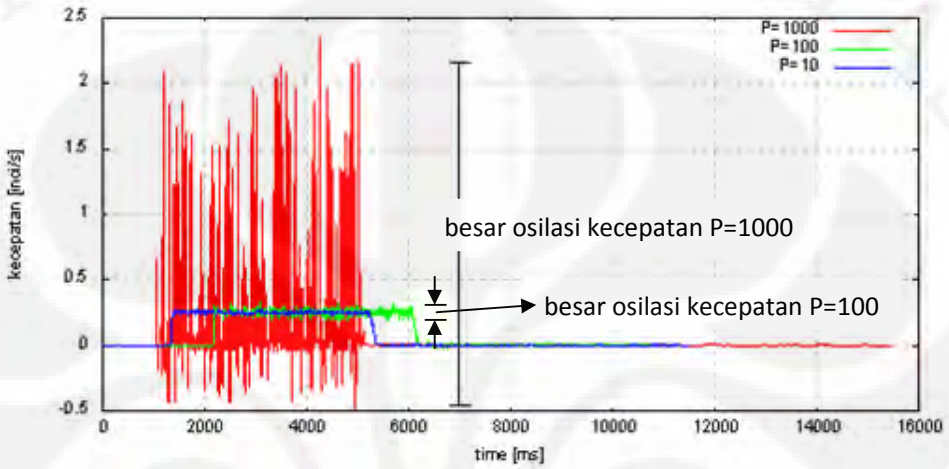
1. Tiap pengujian dilakukan dengan perintah G-Code G01 X1.0 F10.
2. Memvariasikan parameter deadband pada modul PID EMC2. Nilai deadbeand ditentukan dengan *trial and error*. Nilai variasinya adalah 0.00005 inci dan 0.005 inci.
3. Nilai P dan D pada modul PID EMC2 diset tetap dimana nilai P=100 dan D=0.

Pada Gambar 4.1 terlihat bahwa motor berusaha mempertahan kecepatannya pada kecepatan 0.167 inci per menit. Semakin besar deadband yang diberikan maka osilasi pada kecepatan motor akan semakin kecil. Dengan cara pemberian deadband pada modul PID, bisa mencegah motor berosilasi pada titik

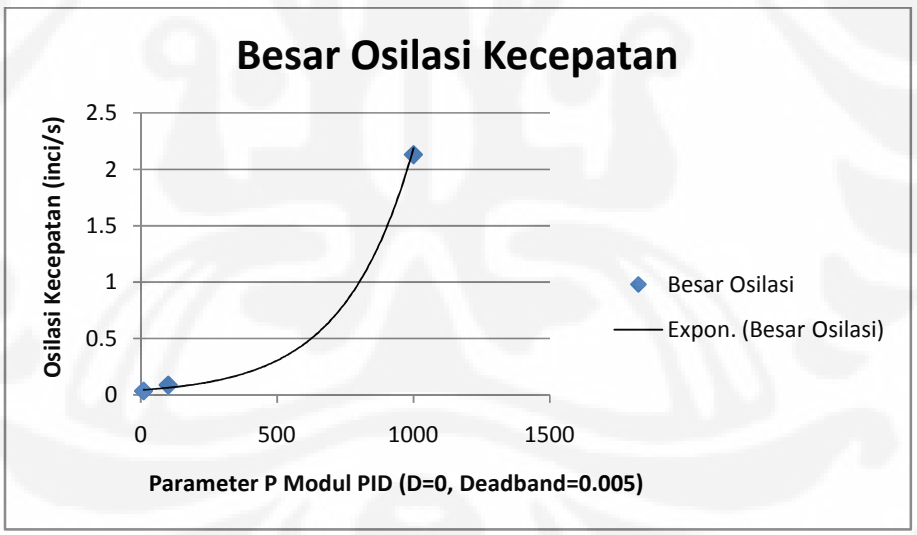

db=0.00005
db=0.005
kecepatan [inci/s]
time [ms]
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
2000
4000
6000
8000
10000
12000
14000


command_P=10
feedback_P=10
command_P=100
feedback_P=100
command_P=1000
feedback_P=1000
value
time [ms]
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2000
4000
6000
8000
10000
12000
14000
16000

kecepatan untuk servopack. Semakin stabil error posisi saat menuju posisi tertentu maka kecepatan motor akan semakin stabil. Saat parameter Kp semakin besar, maka ouput ke servopack dari modul PID akan semakin berfluktuatif (Gambar 4.5). Akibat tegangan referensi yang berfluktuatif pada servopack, sistem pada servopack akan berusaha mengikuti perubahan kecepatan tersebut (Gambar 4.6).

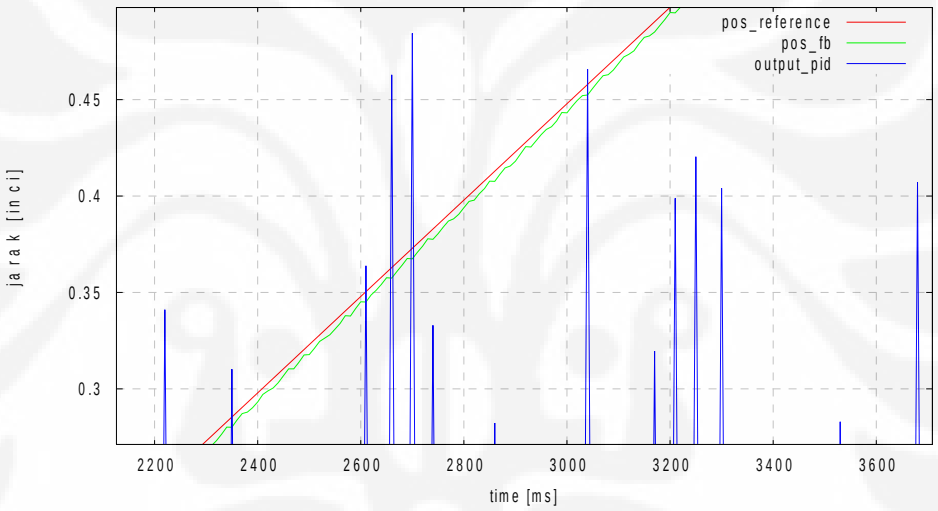


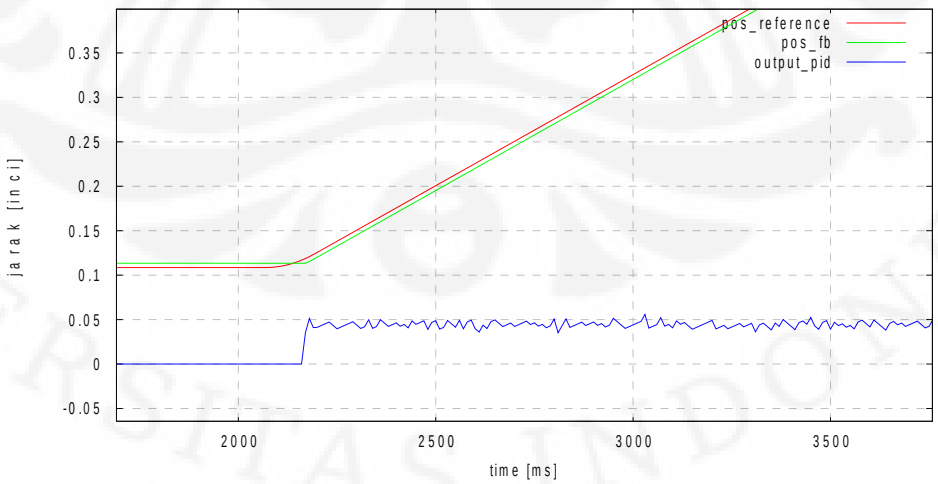
Gambar 4.3. Grafik perbandingan kecepatan terhadap variasi parameter P

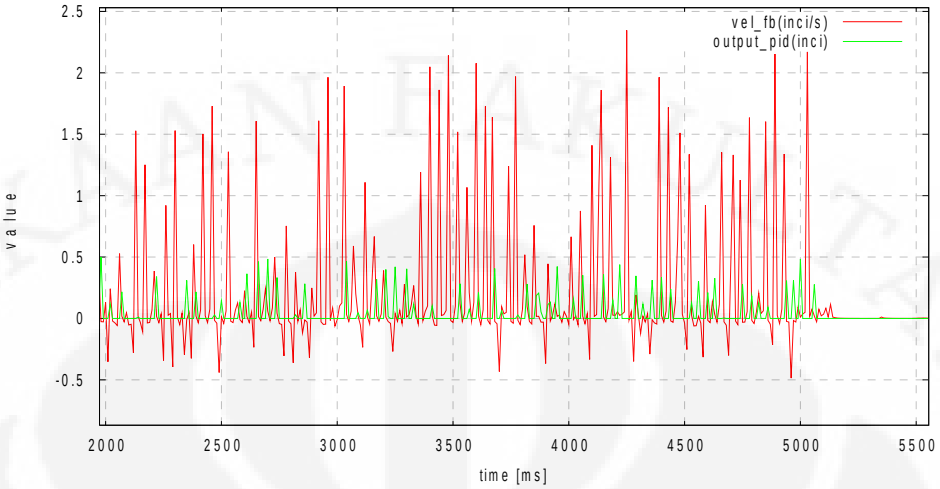


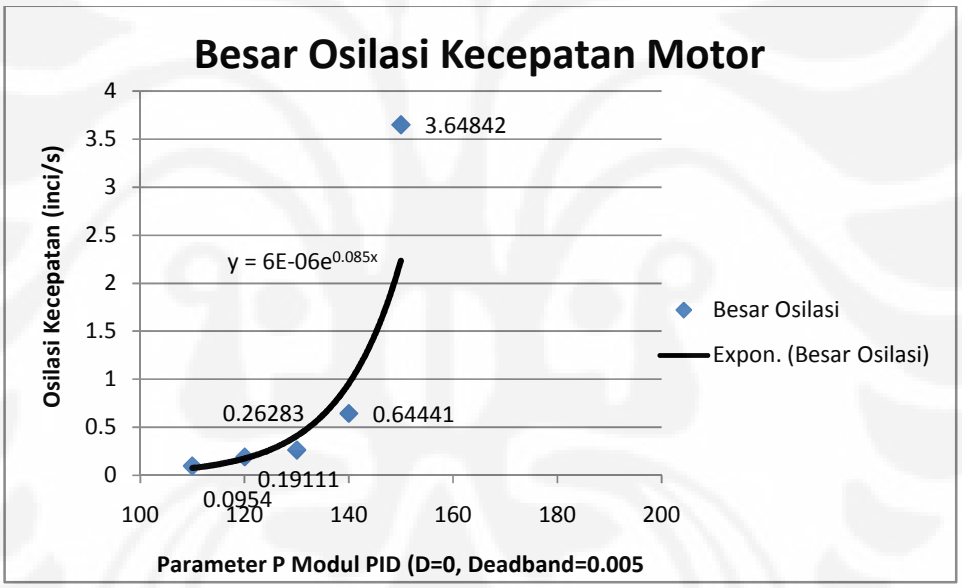
Gambar 4.4 Grafik perubahan besar osilasi kecepatan terhadap variasi parameter P

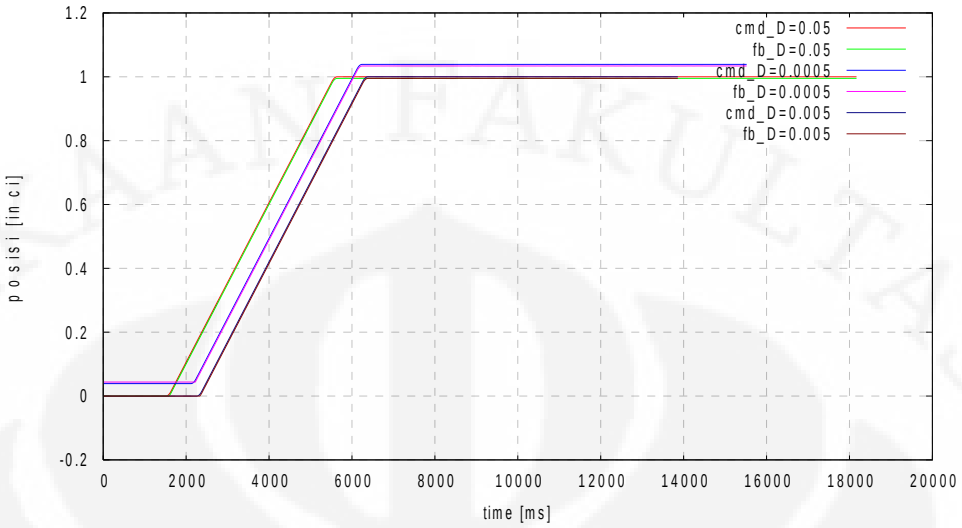
Besarnya kecepatan pada motor ditentukan oleh besarnya error pada modul PID EMC2. Semakin besar kecepatan yang diinginkan maka error yang terjadi saat menuju posisi akan dibuat semakin besar dengan memperbesar perubahan set point pada modul PID EMC2. Seiring dengan pertambahan kecepatan motor, maka osilasi dari kecepatan motor yang terjadi akan semakin

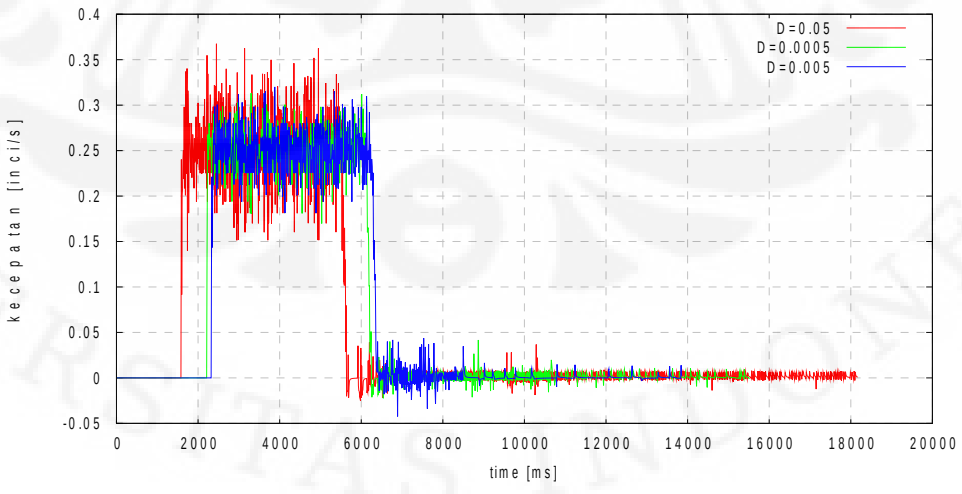

pos_reference
pos_fb
output_pid
jarak [inci]
0.45
0.4
0.35
0.3
2200
2400
2600
2800
3000
3200
3400
3600
time [ms]

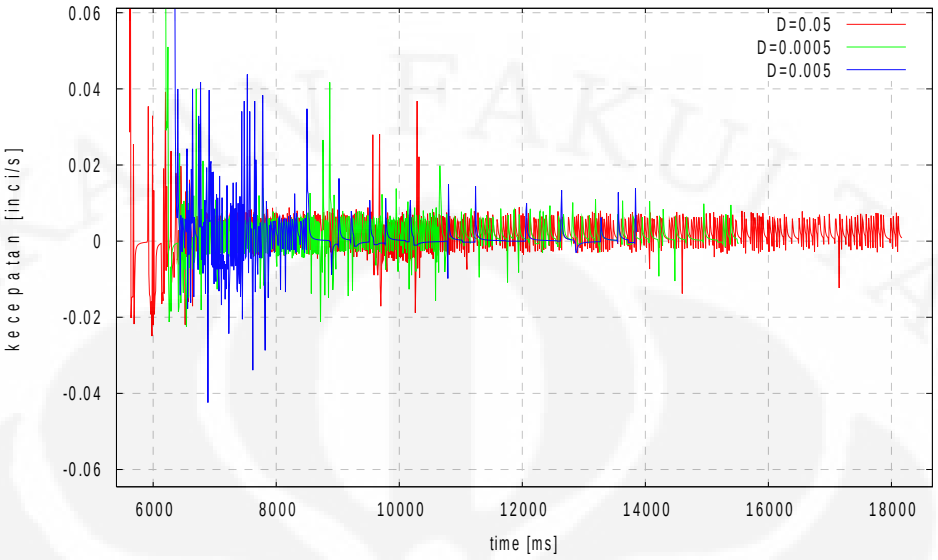

pos_reference
pos_fb
output_pid
jarak [inci]
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
2000
2500
3000
3500
time [ms]

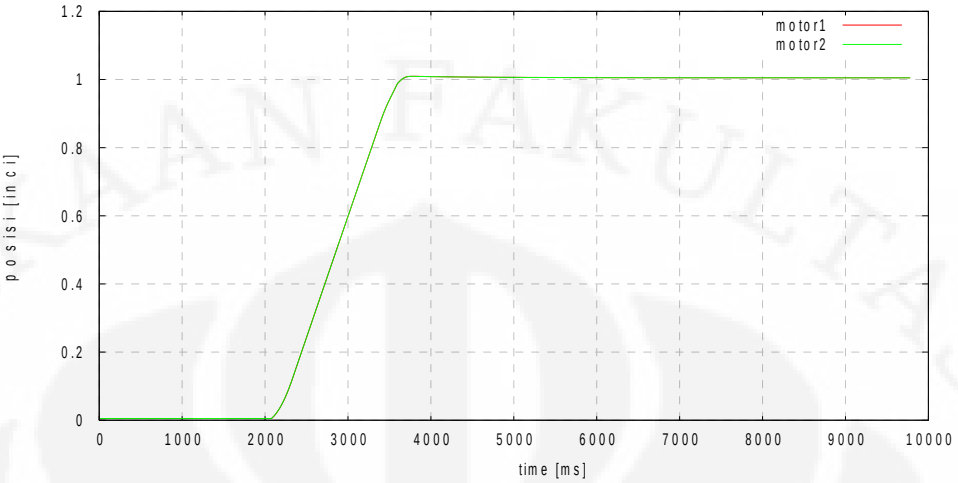

vel_fb(inci/s)
output_pid(inci)
value
time [ms]
2.5
2
1.5
1
0.5
0
-0.5
2000
2500
3000
3500
4000
4500
5000
5500

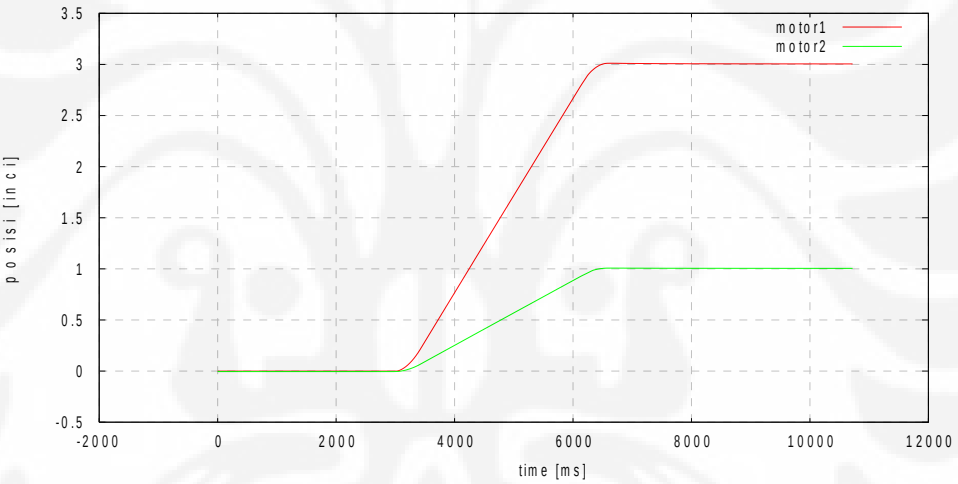

Besar Osilasi Kecepatan Motor
Osilasi Kecepatan (inci/s)
y = 6E-06e0.085x
3.64842
0.64441
0.26283
0.19111
0.0954
Besar Osilasi
Expon. (Besar Osilasi)
Parameter P Modul PID (D=0, Deadband=0.005
4
3.5
3
2.5
2
1.5
1
0.5
0
100
120
140
160
180
200


cmd_D=0.05
fb_D=0.05
cmd_D=0.0005
fb_D=0.0005
cmd_D=0.005
fb_D=0.005
posisi [inci]
time [ms]
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000


D=0.05
D=0.0005
D=0.005
kecepatan [inci/s]
time [ms]
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000


D=0.05
D=0.0005
D=0.005
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
kecepatan [inci/s]
6000
8000
10000
12000
14000
16000
18000
time [ms]


motor1
motor2
posisi [inci]
time [ms]
1.2
1
0.8
0.6
0.4
0.2
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


motor1
motor2
posisi [inci]
time [ms]
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-2000
0
2000
4000
6000
8000
10000
12000

BAB 5
KESIMPULAN

Berdasarkan hasil pengujian dan analisa data, maka dapat ditarik beberapa kesimpulan, yaitu :

1. EMC2 bekerja dengan memperbaharui set point setiap periode servo agar bisa mengatur total waktu yang dibutuhkan untuk satu perintah.
2. EMC2 telah dapat diimplementasikan pada mesin CNC.
3. Performansi mesin tidak hanya ditentukan oleh besar kecilnya error posisi yang terjadi tetapi ditentukan oleh kestabilan kecepatan motor yang digunakan untuk mencapai suatu posisi.
4. Semakin besar parameter P pada modul PID maka kecepatan motor semakin tidak stabil.
5. Nilai P yang diperbolehkan untuk mesin CNC yang dirancang adalah 130.
6. Kecepatan motor akan lebih stabil jika menggunakan parameter D tepat pada modul PID.

DAFTAR REFERENSI

[1] Mesa. (n.d.). *5i20Anything I/O Manual*. November 21, 2009. http://www.mesanet.com

[2] Stenerson J., & Curran K. (1997). *Computer Numerical Control Operation and Programming*. Ohio : Prentice Hall.

[3] The EMC Team (2009, July 31). *Integrator Manual V2.3*. September 16, 2009. http://www.linuxcnc.org

[4] The EMC Team (2009, September 12). *User Manual V2.3*. September 16, 2009. http://www.linuxcnc.org

```
[HOSTMOT2]
DRIVER=hm2_pci
BOARD=5i20
CONFIG="firmware=hm2/5i20/SVST8_4.BIT num_encoders=3
num_pwmgens=3 num_stepgens=0"

[EMC]
# Name of machine, for use with display, etc.
MACHINE =              HM2-Servo
# Name of NML file to use, default is emc.nml
NML_FILE =             emc.nml
# Debug level, 0 means no messages. See src/emc/nml_int/emcglb.h for others
#DEBUG =               0x00000003
#DEBUG =               0x00000007
DEBUG = 0

[DISPLAY]
# Name of display program, e.g., tkemc
DISPLAY =              axis
# Cycle time, in seconds, that display will sleep between polls
CYCLE_TIME =           0.0500
# Path to help file
HELP_FILE =            tkemc.txt
# Initial display setting for position, RELATIVE or MACHINE
POSITION_OFFSET =      RELATIVE
# Initial display setting for position, COMMANDED or ACTUAL
POSITION_FEEDBACK =    ACTUAL
# Highest value that will be allowed for feed override, 1.0 = 100%
MAX_FEED_OVERRIDE =    1.5
# Prefix to be used
PROGRAM_PREFIX = /home/kendali/emc2/nc_files
# Introductory graphic
INTRO_GRAPHIC =        emc2.gif
INTRO_TIME =           5

[TASK]
# Name of task controller program, e.g., milltask
TASK =                 milltask
# Cycle time, in seconds, that task controller will sleep between polls
CYCLE_TIME =           0.010
```

(Lanjutan)

```
[RS274NGC]
# File containing interpreter variables
PARAMETER_FILE = hm2-servo.var

[EMCMOT]
EMCMOT =              motmod
# Servo task period, in nanoseconds
SERVO_PERIOD =        1000000

[HAL]
# The run script first uses halcmd to execute any HALFILE
# files, and then to execute any individual HALCMD commands.
# list of hal config files to run through halcmd
# files are executed in the order in which they appear
HALFILE = hm2-servo.hal
# list of halcmd commands to execute
# commands are executed in the order in which they appear
#HALCMD =             save neta

[TRAJ]
AXES =                3
# COORDINATES =       X Y Z R P W
COORDINATES =         X Y Z
HOME =                0 0 0
LINEAR_UNITS =        inch
ANGULAR_UNITS =       degree
CYCLE_TIME =          0.001
DEFAULT_VELOCITY =    3.0
MAX_VELOCITY =        4.0
DEFAULT_ACCELERATION =  6.0
MAX_ACCELERATION =    7.0
#POSITION_FILE = position.txt
NO_FORCE_HOMING = 1

[AXIS_0]
TYPE =                LINEAR
HOME =                0.000
MAX_VELOCITY =        4.0
MAX_ACCELERATION =    4.0
BACKLASH = 0.000
INPUT_SCALE =         8670
```

(Lanjutan)

```
OUTPUT_SCALE = -1.000
OUTPUT_OFFSET =        0.0
MIN_LIMIT =           -3.0
MAX_LIMIT =           32.0
FERROR = 1.000
MIN_FERROR = 0.500
MAX_OUTPUT =          1.0
# PID tuning params
DEADBAND =            0.00500
P =                   100.0
I =                   0.000
D =                   0.0050
FF0 =                 0.000
FF1 =                 0.000
FF2 =                         0.0
BIAS =                0.000

[AXIS_1]
TYPE =                LINEAR
HOME =                0.000
MAX_VELOCITY =        4.0
MAX_ACCELERATION =    4.0
BACKLASH = 0.000
INPUT_SCALE =         4960
OUTPUT_SCALE = -1.000
OUTPUT_OFFSET =       0.0
MIN_LIMIT =           -3.0
MAX_LIMIT =           32.0
FERROR = 1.000
MIN_FERROR = 0.500
MAX_OUTPUT =          1.0
# PID tuning params
DEADBAND =            0.00500
P =                   100.0
I =                   0.000
D =                   0.0050
FF0 =                 0.000
FF1 =                 0.000
FF2 =                         0.0
BIAS =                0.000
```

(Lanjutan)

```
[AXIS_2]
TYPE =                LINEAR
HOME =                0.000
MAX_VELOCITY =        4.0
MAX_ACCELERATION =    4.0
BACKLASH = 0.000
INPUT_SCALE =         -8670
OUTPUT_SCALE = 1.000
OUTPUT_OFFSET =       0.0
MIN_LIMIT =           -3.0
MAX_LIMIT =           32.0
FERROR = 1.000
MIN_FERROR = 0.500
MAX_OUTPUT =          1.0
# PID tuning params
#DEADBAND =           0.000015
DEADBAND =            0.00500
P =                   100.0
I =                   0.000
D =                   0.0050
FF0 =                 0.000
FF1 =                 0.000
FF2 =                       0.0
BIAS =                0.000

[EMCIO]
# Name of IO controller program, e.g., io
EMCIO =               io
# cycle time, in seconds
CYCLE_TIME =          0.100
# tool table file
TOOL_TABLE = tool.tbl
```

```
# kinematics
loadrt trivkins

# motion controller, get name and thread periods from
     ini file
loadrt [EMCMOT]EMCMOT
     servo_period_nsec=[EMCMOT]SERVO_PERIOD
     num_joints=[TRAJ]AXES
# standard components
loadrt pid num_chan=3
# hostmot2 driver
loadrt hostmot2
# load low-level driver
loadrt [HOSTMOT2](DRIVER) config=[HOSTMOT2](CONFIG)

setp hm2_[HOSTMOT2](BOARD).0.pwmgen.pdm_frequency
     6000000
setp hm2_[HOSTMOT2](BOARD).0.watchdog.timeout_ns
     10000000

# ################################################
# THREADS
# ################################################
addf hm2_[HOSTMOT2](BOARD).0.read         servo-thread
addf motion-command-handler               servo-thread
addf motion-controller                    servo-thread
addf pid.0.do-pid-calcs                   servo-thread
addf pid.1.do-pid-calcs                   servo-thread
addf pid.2.do-pid-calcs                   servo-thread
addf hm2_[HOSTMOT2](BOARD).0.write        servo-thread
addf hm2_[HOSTMOT2](BOARD).0.pet_watchdog  servo-thread
```

(Lanjutan)

```
# ################
# X [0] Axis
# ################

# axis enable chain
newsig emcmot.00.enable bit
sets emcmot.00.enable FALSE
net emcmot.00.enable => pid.0.enable
net emcmot.00.enable =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.00.enable
net emcmot.00.enable <= axis.0.amp-enable-out

# encoder feedback
setp hm2_[HOSTMOT2](BOARD).0.encoder.00.counter-mode 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.00.filter 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.00.index-invert 0
setp hm2_[HOSTMOT2](BOARD).0.encoder.00.index-mask 0
setp hm2_[HOSTMOT2](BOARD).0.encoder.00.index-mask-
     invert 0
setp  hm2_[HOSTMOT2](BOARD).0.encoder.00.scale
     [AXIS_0](INPUT_SCALE)
net motor.00.pos-fb
     hm2_[HOSTMOT2](BOARD).0.encoder.00.position =>
     pid.0.feedback
net motor.00.pos-fb => axis.0.motor-pos-fb #push copy
     back to Axis GUI

# set PID loop gains from inifile
setp pid.0.Pgain [AXIS_0]P
setp pid.0.Igain [AXIS_0]I
setp pid.0.Dgain [AXIS_0]D
setp pid.0.bias [AXIS_0]BIAS
```

(Lanjutan)

```
setp pid.0.FF0 [AXIS_0]FF0
setp pid.0.FF1 [AXIS_0]FF1
setp pid.0.FF2 [AXIS_0]FF2
setp pid.0.deadband [AXIS_0]DEADBAND
setp pid.0.maxoutput [AXIS_0]MAX_VELOCITY

# position command signals
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.00.output-type 3
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.00.scale
     [AXIS_0](OUTPUT_SCALE)
net emcmot.00.pos-cmd axis.0.motor-pos-cmd =>
     pid.0.command
net motor.00.command  pid.0.output  =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.00.value


# ################
# Y [1] Axis
# ################

# axis enable chain
newsig emcmot.01.enable bit
sets emcmot.01.enable FALSE
net emcmot.01.enable => pid.1.enable
net emcmot.01.enable =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.01.enable
net emcmot.01.enable <= axis.1.amp-enable-out

# encoder feedback
setp hm2_[HOSTMOT2](BOARD).0.encoder.01.counter-mode 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.01.filter 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.01.index-invert 0
```

(Lanjutan)

```
setp hm2_[HOSTMOT2](BOARD).0.encoder.01.index-mask 0
setp hm2_[HOSTMOT2](BOARD).0.encoder.01.index-mask-
     invert 0
setp  hm2_[HOSTMOT2](BOARD).0.encoder.01.scale
     [AXIS_1]INPUT_SCALE
net motor.01.pos-fb
     hm2_[HOSTMOT2](BOARD).0.encoder.01.position =>
     pid.1.feedback
net motor.01.pos-fb => axis.1.motor-pos-fb #push copy
     back to Axis GUI

# set PID loop gains from inifile
setp pid.1.Pgain [AXIS_1]P
setp pid.1.Igain [AXIS_1]I
setp pid.1.Dgain [AXIS_1]D
setp pid.1.bias [AXIS_1]BIAS
setp pid.1.FF0 [AXIS_1]FF0
setp pid.1.FF1 [AXIS_1]FF1
setp pid.1.FF2 [AXIS_1]FF2
setp pid.1.deadband [AXIS_1]DEADBAND
setp pid.1.maxoutput [AXIS_1]MAX_OUTPUT

# position command signals
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.01.output-type 3
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.01.scale
     [AXIS_1]OUTPUT_SCALE
net emcmot.01.pos-cmd axis.1.motor-pos-cmd =>
     pid.1.command
net motor.01.command  pid.1.output  =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.01.value
```

(Lanjutan)

```
# ################
# Z [2] Axis
# ################

# axis enable chain
newsig emcmot.02.enable bit
sets emcmot.02.enable FALSE
net emcmot.02.enable => pid.2.enable
net emcmot.02.enable =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.02.enable
net emcmot.02.enable <= axis.2.amp-enable-out

# encoder feedback
setp hm2_[HOSTMOT2](BOARD).0.encoder.02.counter-mode 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.02.filter 1
setp hm2_[HOSTMOT2](BOARD).0.encoder.02.index-invert 0
setp hm2_[HOSTMOT2](BOARD).0.encoder.02.index-mask 0
setp hm2_[HOSTMOT2](BOARD).0.encoder.02.index-mask-
     invert 0
setp  hm2_[HOSTMOT2](BOARD).0.encoder.02.scale
     [AXIS_2]INPUT_SCALE
net motor.02.pos-fb
     hm2_[HOSTMOT2](BOARD).0.encoder.02.position =>
     pid.2.feedback
net motor.02.pos-fb => axis.2.motor-pos-fb #push copy
     back to Axis GUI

# set PID loop gains from inifile
setp pid.2.Pgain [AXIS_2]P
setp pid.2.Igain [AXIS_2]I
setp pid.2.Dgain [AXIS_2]D
setp pid.2.bias [AXIS_2]BIAS
```

(Lanjutan)

```
setp pid.2.FF0 [AXIS_2]FF0
setp pid.2.FF1 [AXIS_2]FF1
setp pid.2.FF2 [AXIS_2]FF2
setp pid.2.deadband [AXIS_2]DEADBAND
setp pid.2.maxoutput [AXIS_2]MAX_OUTPUT

# position command signals
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.02.output-type 3
setp hm2_[HOSTMOT2](BOARD).0.pwmgen.02.scale
     [AXIS_2]OUTPUT_SCALE
net emcmot.02.pos-cmd axis.2.motor-pos-cmd =>
     pid.2.command
net motor.02.command  pid.2.output  =>
     hm2_[HOSTMOT2](BOARD).0.pwmgen.02.value

# ################################################
# Standard I/O Block - EStop, Etc
# ################################################

# create a signal for the estop loopback
net estop-loop iocontrol.0.user-enable-out =>
     iocontrol.0.emc-enable-in

# create signals for tool loading loopback
net tool-prep-loop iocontrol.0.tool-prepare =>
     iocontrol.0.tool-prepared
net tool-change-loop iocontrol.0.tool-change =>
     iocontrol.0.tool-changed
```