TUGAS AKHIR – TE 145561

SISTEM KONTROL DAN *HMI* PADA MODUL ELEKTRO HIDROLIK

Fajar Alif Chalifatullah
NRP 10311500000005

Dosen Pembimbing
Slamet Budi Prayitno, ST., MT

PROGRAM STUDI KOMPUTER KONTROL
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – **TE 145561**

CONTROL SYSTEM AND HMI ON ELECTRIC HYDRAULIC MODULE

Fajar Alif Chalifatullah
NRP 10311500000005

Advisor
Slamet Budi Prayitno, ST., MT

COMPUTER CONTROL STUDY PROGRAM
Electrical and Automation Engineering Department
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2018

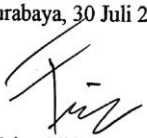
PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Sistem Kontrol Dan *HMI* Pada Modul Elektro Hidrolik**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 30 Juli 2018



Fajar Alif Chalifatullah
NRP 10311500000005

-----Halaman ini sengaja dikosongkan-----

SISTEM KONTROL DAN *HMI* PADA MODUL ELEKTRO HIDROLIK

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing

Slamet Budi Prayitno, ST., MT
NIP. 197811113201021002

SURABAYA
JULI , 2018

“Halaman ini sengaja dikosongkan”

ABSTRAK

Sistem hidrolik merupakan suatu bentuk perubahan atau pemindahan daya dengan menggunakan media penghantar berupa fluida cair untuk memperoleh daya yang lebih besar dari daya awal yang dikeluarkan. Modul Elektro Hidrolik di Departemen Teknik Elektro Otomasi digunakan untuk kontrol kecepatan menggunakan *Servo Valve*. Modul ini bekerja dengan sistem kontrol *loop* tertutup. Kontroler pada modul digunakan untuk mengatur buka dan tutup katub pada *Servo Valve* yang digunakan untuk mengatur berapa tekanan oli yang dibutuhkan untuk mencapai kecepatan yang diinginkan. Saat ini kontroler pada modul Elektro Hidrolik sedang mengalami kerusakan dan juga masih belum memiliki *HMI (Human Machine Interface)*.

Tugas akhir ini membuat *HMI (Human Machine Interface)* dan sistem kontrol pada modul Elektro Hidrolik. *HMI (Human Machine Interface)* yang di bangun menggunakan *software Visual Basic* yang di hubungkan menggunakan *USB*. Sistem kontrolnya menggunakan kontroler *PID* untuk mengkontrol kecepatan pada motor hidrolik dengan mengatur *set point* yang diinginkan dan mengatur nilai Kp, Ki, dan Kd yang dibutuhkan agar *output* sesuai dengan yang diinginkan.

HMI (Human Machine Interface) dapat menampilkan semua data dari keadaan awal mulai yaitu keadaan nol sampai sistem mencapai *setpoint* yang diinginkan dalam bentuk grafik. Data yang diperoleh dapat disimpan dalam bentuk *file excel* dan siap untuk diolah lagi. Kontroler *PID* dengan Kp = 10, Ki = 1, Kd = 28 dapat mempercepat *rise time* untuk mencapai *setpoint*. Sehingga hanya butuh 1,5 detik untuk mencapai *setpoint* di 100 *RPM*. Sedangkan jika tanpa kontroler waktu yang dibutuhkan sebesar 6 detik baru bisa mencapai *setpoint.*

Kata Kunci __ Kontrol Kecepatan, Modul Hidrolik, Kontrol *PID.*

“Halaman Ini Sengaja Dikosongkan”

ABSTRACT

Hydraulic system is a change or transfer of power by using fluid medium to obtain power greater than the initial power released. Hydraulic Electro Module in the Department of Electrical Automation Engineering is used for speed control using Servo Valve. This module works with a closed-loop control system. The controller on the module is used to adjust the valve open and close on the Servo Valve which is used to adjust the oil pressure required to achieve the desired speed. at this time the controller on the Electro Hydraulic module is being damaged and also does not have HMI (Human Machine Interface).

This final project makes HMI (Human Machine Interface) and control system on Electro Hydraulic module. HMI (Human Machine Interface) is built using Visual Basic software which connected using USB. For the control system use PID controller to control the speed of the hydraulic motor by setting the desired set point and set the value of Kp, Ki, and Kd which required for the output as desired.

HMI (Human Machine Interface) can display all data from initial state starting from zero until the system reaches the desired setpoint in graphical form. The data can be stored in the form of excel file and ready to be processed again. The PID controller with Kp = 10, Ki = 1, Kd = 28 can increase speed of rise time to reach the setpoint. so it only takes 1.5 seconds to reach the setpoint at 100 RPM. But if no controller, need 6 seconds of time to reach setpoint.

Keywords __ Speed Control, Hydraulic Module, PID Control

“Halaman Ini Sengaja Dikosongkan”

KATA PENGANTAR

Alhamdulillah, segala puji syukur bagi Allah SWT karena berkat rahmatNya, penulis dapat menyelesaikan Proyek Akhir yang berjudul :

"SISTEM KONTROL DAN HMI PADA MODUL ELEKTRO HIDROLIK"

Pembuatan dan penyusunan Proyek Akhir ini diajukan sebagai salah satu syarat untuk menyelesaikan Studi Diploma III (D3) guna memperoleh Gelar Ahli Madya (A.Md) di Departemen Teknik Elektro Otomasi Institut Teknologi Sepuluh Nopember Surabaya.

Penulis berusaha dengan sekuat badan dan hati, segenap jiwa dan raga dengan segala keterbatasan, kebodohan dan kekurangannya sehingga dapat diselesaikannya laporan dan pengerjaan Proyek Akhir ini oleh karena atas berkat Rahmat Allah yang maha kuasa dan dengan di dorong oleh keinginan yang luhur serta banyak dibantu oleh berbagai pihak. Tak lupa penulis memohon ampunan dan maaf atas seluruh kesalahan yang selalu penulis perbuat.

Demikian besar harapan penulis, semoga laporan Proyek Akhir ini dapat bermanfaat bagi yang mau peduli ataupun tidak.

Surabaya, 30 Juli 2018

Penulis

----“Halaman ini sengaja dikosongkan”----

UCAPAN TERIMA KASIH

Kami sangat bersyukur atas berkat rahmatNya, hingga saat ini kami masih diberikan kesempatan untuk dapat mengenyam pendidikan yang tinggi sehingga kami dapat menyelesaiakn laporan serta pengerjaan Proyek Akhir ini. Kami haturkan doa dan rasa terimakasih ini, khususnya kepada :

1. Beliau Baginda Rosulalloh SAW, Waliullah, dan seluruh kekasih – kekasihNya.
2. Kedua Orang Tua dan keluarga kami yang selalu dan setiap saat memberikan dukungan penuh.
3. Bapak Slamet Budiprayitno, ST., MT. selaku dosen pembimbing Proyek Akhir dari penulis serta Bapak dan Ibu Dosen penguji.
4. Seluruh Bapak dan Ibu dosen dan karyawan yang telah membimbing dan membekali ilmu kepada penulis selama menempuh pendidikan di kampus Institut Teknologi Sepuluh Nopember (ITS).
5. Seluruh Bapak, Ibu, mahasiswa ITS dan pihak – pihak yang terkait dengan kampus ITS, khususnya Departemen Teknik Elektro Otomasi yang telah membantu dan memberikan dukungan langsung maupun tidak langsung atas terselesaikannya Tugas Akhir ini.
6. Seluruh keluarga HYDRA'15 angkatan 2015 Departemen Teknik Elektro Otomasi yang sudah saling membantu dan memberi semangat untuk menyelesaikan tugas akhir ini.
7. Untuk seluruh keluarga angkatan 2016 Departemen Teknik Elektro Otomasi yang sudah ikut membantu dalam pengambilan data dengan menyediakan alat – alat didalam Laboratorium saat proses pengambilan data.
8. Tak lupa bagi Pembaca Buku Proyek Akhir ini.

Besar harapan kami semoga Allah SWT selalu melimpahkan rahmatNya kepada seluruh pihak yang telah membantu terselesaikannya pengerjaan dan pembuatan laporan Proyek Akhir ini serta turut membantu pula dalam perkembangan kami selama studi di kampus ini untuk menjadikan kami insan yang berguna bagi sesama.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

----Halaman ini sengaja dikosongkan----

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Sistem Pneumatik dan Hidrolik telah digunakan selama bertahun - tahun dalam proses industri dan dengan demikian telah memperoleh tempat yang mapan di industri modern. Perkembangan teknologi cairan fluida terus-menerus selama bertahun-tahun secara signifikan telah memperluas dan meningkatkan aplikasi ke banyak wilayah untuk mengadopsi teknologi Pneumatik dan Hidrolik. Beberapa penggunaan utama teknologi tenaga fluida adalah industri manufaktur terutama industri otomotif, produsen alat mesin serta produsen alat rumah tangga dan komersial. Didalam sistem Hidrolik juga dibutuh kan suatu sistem kontrol yang digunakan untuk mengatur gerak dari hidrolik tersebut. Sistem kontrol merupakan suatu kumpulan cara atau metode yang dipelajari dari kebiasaan-kebiasaan manusia dalam bekerja, dimana manusia membutuhkan suatu pengamatan kualitas dari apa yang telah mereka kerjakan sehingga memiliki karakteristik sesuai dengan yang diharapkan pada mulanya. Perkembangan teknologi menyebabkan manusia selalu terus belajar untuk mengembangkan dan mengoperasikan pekerjaan-pekerjaan kontrol yang semula dilakukan oleh manusia menjadi serba otomatis (dikendalikan oleh mesin). Dalam aplikasinya, sistem kontrol memegang peranan penting dalam teknologi. Sebagai contoh, otomatisasi industri dapat menekan biaya produksi, mempertinggi kualitas, dan dapat menggantikan pekerjaan-pekerjaan rutin yang membosankan. Sehingga dengan demikian akan meningkatkan kinerja suatu sistem secara keseluruhan, dan pada akhirnya memberikan keuntungan bagi manusia yang menerapkannya. Sistem kontrol didalam suatu *plant* membutuhkan sebuah *software* yang bisa digunakan untuk merepresesntasikan data dari suatu sistem yaitu melalui *HMI (Human Machine Interface)*. *HMI (Human Machine Interface)* adalah sebuah *interface* atau tampilan penghubung antara manusia dengan mesin. *HMI* juga merupakan *user interface* dan sistem kontrol untuk manufaktur.

Departemen Teknik Elektro Otomasi mempunyai sebuah modul Elektro Hidrolik yang digunakan untuk kontrol kecepatan yang aktuatornya menggunakan *Servo Valve*. Modul ini bekerja dengan sistem kontrol *loop* tertutup. Kontroler pada modul digunakan untuk mengatur buka dan tutup katub pada *Servo Valve* yang digunakan untuk mengatur berapa tekanan oli yang dibutuhkan untuk mencapai kecepatan yang diinginkan. Tetapi untuk saat ini kontroler pada modul Elektro Hidrolik sedang mengalami kerusakan dan juga modul ini masih belum memiliki *HMI (Human Machine Interface)*.

Sehingga Untuk tugas akhir ini kami berencana untuk meretrofit *HMI (Human Machine Interface)* dan sistem kontrol pada modul elektro hidrolik. *HMI (Human Machine Interface)* yang akan kami bangun menggunkan software *Visual Basic* yang di *interface* menggunakan *USB*. *HMI (Human Machine Interface)* dan sistem kontrol *loop* tertutup diharapkan dapat memudahkan praktikan dalam pengambilan data saat melakukan praktikum dengan menggunakan modul hidrolik ini.

1.2 Tujuan Dan Manfaat

Tujuan yang akan dicapai dari Tugas Akhir ini adalah sebagai berikut:

1. Merepresentasikan semua data praktikum yang sudah diambil melalui *HMI* dengan *software Visual Basic*.
2. Membuat sistem kontrol yang dapat mempercepat *rise time* dan memepertahankan nilai kecepatan motor Hidrolik.

1.3 Perumusan Masalah

Dari uraian singkat diatas, maka permasalahan yang akan dibahas dalam perancangan Proyek Akhir ini adalah sebagai berikut:

1. Mengkontrol *Servo Valve* tidak bisa langsung menggunakan *output* dari *DAC* karena arusnya *output*-nya terlalu kecil.
2. Menampilkan dan mengirimkan data dari *HMI* ke *mikrokontroler* atau sebaliknya, terlebih dahulu harus melakukan *parsing* data agar data dapat tampil dengan runtut dan benar.

1.4 Batasan Masalah

Batasan masalah dari pengerjaan Proyek Akhir ini adalah:

1. Mekanik pada sistem hidrolik diabaikan.
2. Menggunakan kontroler *PID* untuk kontrol kecepatan pada modul hidrolik.
3. Menggunakan modul hidrolik yang sudah ada di Departemen Teknik Elektro Otomasi.
4. Tidak membahas Tekanan fluida yang digunakan pada modul hidrolik.
5. *HMI* menggunakan *software visual basic*.
6. Menggunakan sensor kecepatan yang sudah tersedia pada modul Elektro Hidrolik.

1.5 Metodologi

Dalam pelaksanaan tugas akhir berupa pembuatan Sistem Kontrol dan *HMI (Human Machine Interface)* Pada Modul Elektro Hidrolik Ada beberapa tahap yang perlu dipersiapkan yaitu sebagai berikut:

1. Pengamatan Masalah

Pada kegiatan ini penulis mendalami latar belakang permasalahan dan mengamati keadaan terkini terkait permasalahan sistem kontrol untuk hidrolik terutama untuk mengkontrol *Servo Valve*. permasalahan koneksi antara *HMI (Human Machine Interface)* dan STM32F103C. Serta masalah – masalah teknis yang terjadi pada hidrolik.

2. Studi Literatur

Studi literatur merupakan tahap pencarian data dan literatur dari sumber – sumber yang dapat di pertanggung jawabkan kebenarannya. Disini sumber bisa diperoleh dari jurnal, *paper*, buku ilmiah, *manual book* dan *ebook* yang didapat dari internet. Literatur yang dicari untuk Tugas Akhir ini adalah mengenai Sistem kontrol hidrolik, cara mengkontrol *Servo Valve*, cara koneksi antara *HMI* dan STM32F103C, serta pembuatan *HMI (Human Machine Interface)* menggunakan *Software Visual Basic*.

3. Perancangan Alat

Pada tugas akhir ini saya tidak membuat alat berupa *prototype* karena alat berupa modul elektro hidrolik sudah tersedia di Departemen Teknik Elektro Otomasi. Saya hanya meretrofit pada bagian system kontrolnya dan menambahakan *HMI (Human Machine Interface)* pada modul elektro hidrolik tersebut. *Mikrokontroler* yang digunakan adalah STM32F103 dengan *ADC 12 bit* agar pengiriman datanya menjadi lebih akurat. STM32F103C ini digunakan untuk mengatur buka tutup *Servo Valve* agar oli yang keluar sesuai dengan yang dibutuhkan. Nantinya STM32F103C tersebut agar bisa mengatur *Servo Valve* akan disambungkan ke *Driver Servo Valve* tersebut. *HMI (Human Machine Interface)* yang dibangun memenggunakan *Software Visual Basic* dan menggunakan koneksi USB. Didalam *HMI (Human Machine Interface)* terdapat grafik yang berisi respon waktu untuk mencapai set point yang diinginkan, terdapat tempat kolom untuk mengatur set point yang diinginkan, serta disediakan tombol *start* yang digunakan untuk memulai proses pengambilan data dan *stop* digunakan untuk menghentikan proses pengambilan data.

4. Pengujian Alat dan Analisis Data

Setelah pembuatan kontroler dan *HMI (Human Machine Interface)* pada modul elektro hidrolik selesai, maka akan dilakukan pengujian. Hal pertama yang diuji adalah kontroler apakah sudah berjalan dengan sempurna dalam mengatur buka tutup *Servo Valve* sesuai kebutuhan. Menguji ke telitian pembacaan sensor kecepatan , serta menguji koneksi dari *HMI (Human Machine Interface)* ke mikrokontroler apakah data yang ditampilkan di *HMI (Human Machine Interface)* sama dengan keadaan nyata pada modul elektro hidrolik.

5. Kesimpulan

Setelah melakukan beberapa tahap dapat ditarik kesimpulan bahwa *Servo Valve* perlu adanya *Driver* untuk bisa di kontrol oleh mikrokontroler. *HMI (Human Machine Interface)* menggunakan *Visual Basic* dapat dikoneksikan ke mikrokontroler menggunakan komunikasi USB dan bisa memudahkan dalam merepresentasikan data respon dari posisi.

6. Penyusunan Laporan

Tahap terakhir yang perlu dilakukan adalah penyusunan laporan akhir yang bertujuan sebagai bukti tertulis bahwa pernah dilakukan penelitian mengenai hal ini.

1.6 Sistematika Laporan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima Bab dengan sistematika sebagai berikut:

Bab I **Pendahuluan**

Bab ini meliputi latar belakang, permasalahan, batasan masalah, tujuan, metodologi penelitian, sistematika laporan, dan relevansi.

Bab II **Teori Dasar**

Bab ini menjelaskan tentang tinjauan pustaka yang mendukung dalam perencanaan dan pembuatan alat.

Bab III **Perancangan Sistem**

Bab ini membahas mengenai perancangan kontroler hardware berupa rangkaian maupun software dan juga HMI pada modul hidrolik.

Bab IV **Simulasi, Implementasi dan Analisis Sistem**

Bab ini memuat hasil simulasi dan implementasi serta analisis dari hasil tersebut.

Bab V **Penutup**

Bab ini berisi kesimpulan dari Tugas Akhir ini dan saran-saran untuk pengembangan alat ini lebih lanjut.

1.7 Relevansi

Tugas akhir dibuat untuk memudahkan pengambilan data praktikum oleh praktikan saat menggunakan modul elektro hidrolik ini. Data respon kecepatan yang diambil menjadi lebih akurat.

“Halaman Ini Sengaja Dikosongkan”

BAB II
TEORI PENUNJANG

Pada bab ini membahas teori dasar dan teori penunjang dari peralatan-peralatan yang digunakan dalam pembuatan Tugas Akhir dengan judul Sistem Kontrol dan *HMI* Pada Modul Elektro Hidrolik.

2.1 *Microsoft Visual Studio* [1]

2.1.1 Percabangan

Perintah percabangan merupakan perintah yang dapat memberikan pilihan terhadap suatu kondisi , program akan menjalankan perintah apabila suatu kondisi memenuhi syarat tertentu. Untuk percabangan ini akan dibahas antara lain:

- ***IF ... Then***

Suatu perintah percabangan yang mempunyai satu percabangan atau satu blok perintah , tergantung nilai yang akan diuji biasanya terdiri dari satu nilai atau syarat. Perintah percabangan ini memiliki dua bentuk penulisan yaitu *singleline* dan *multiline*.

Penulisan percabangan menggunakan *singleline*:

*if[*kondisi*] Then [*Perintah yang akan dieksekusi*]*

contoh penggunaaan seperti pada gambar 2.1.

```
Sub PercabanganSatuBaris()
        Dim nama = " ", alamat As String
        If nama = "rolly" Then alamat = "Padang"
End Sub
```

Gambar 2.1 Penggunaan *IF* pada *visual basic*

Untuk penulisan *Multiline* dapat kita buat seperti kode dibawah ini:

*If [*kondisi*] Then*
    *[*Perintah*]*
    ……...
End If

- **Penggunaan *While***

Merupakan suatu kontrol yang berfungsi untuk melakukan perulangan yang memiliki suatu syarat tertentu, dan akan terus dijalankan selama syarat tersebut terpenuhi dan begitupun sebaliknya jika syarat tidak terpenuhi maka pernyataan tidak akan dijalankan.

Contoh Penulisan *While*..

*While [*Kondisi*]*
[Perintah 1]
[Perintah 2]
……
Increment/decrement
End While

```
Public Sub penggunaan_while()
        Dim angka As Integer = 100
        While angka <= 200
                MsgBox("Angka Sekarang : "+ angka)
                Angka – angka +1
        End While
End Sub
```

Gambar 2.2 Penggunaan *While* pada *visual basic*

2.1.2 *Form*

Form merupakan media interaksi antara pengguna dengan aplikasi yang anda buat. *Form* terbagi atas dua kategori yaitu:

- ***Form Dinamis***

Yaitu *Form* yang bisa dimanipulasi atau diubah bentuk serta disisipi objek kontrol yang berisi perintah – perintah yang di perlukan oleh aplikasi yang akan anda buat, contohnya *windows Form, Console, Librari, WPF, Database Acces SQLServer*, dll. *Form* dinamis merupakan salah satu bentuk dari *Windows Form* yang digunakan untuk menempatkan objek lain diatas nya dan masih bisa dimanipulasi bentuk dan objek tampilannya.

- ***Form Statis***

Yaitu *Form* yang tidak dapat dimanipulasi atau diubah bentuk serta disisipi objek kontrol, *Form* ini hanya dapat dipanggil dari perintah kode. Contohnya:

1. *Message Box*

   Merupakan *form* yang bertugas untuk menampilkan pesan keterangan terhadap suatu kejadian yang diterima oleh aplikasi Anda. Cara pemanggilan *messagebox* adalah dengan menggunakan koding seperti gambar 2.3.

```
MessageBox.Show("Isi Pesan Yang Ditampilkan
")/Caption Pesan".
MessageBoxButtons. YesNo,MessageBoxIcon.(question).
```

Gambar 2.3 Kode Program *MessageBox*

Penjelasan Kode, untuk menampilkan form pesan singkat yaitu dengan membuatkan kode *MessageBox.show* dimana ada empat parameter yangharus diisi yaitu :

a. Isi Pesan Yang akan ditampilkan, dimana anda bisa membuatkan beberapa kalimat yang bisa ditampilkan oleh form ini.
b. ***Title Form***, yaitu teks yang digunakan untuk memberikan judul dari *form* tersebut.
c. ***MessageBoxButtons***, ini merupakan jenis dari tombol - tombol yang akan ada didalam tampilan *form* tersebut.
d. ***MessageBoxIcon***, ini merupakan jenis *icon* atau gambar yang digunakan untuk ditampilkan pada jendela *form* tersebut yang menandakan sebuah pesan itu ditampilkan sebagai pesan *error* atau sebagai konfirmasi atau pertanyaan dan lainnya.

2. *Input Box*

   Input Box merupakan *form* yang digunakan untuk menampilkan jendela *input user*, tetapi *form* ini juga tidak bisa diubah dan disisipi oleh objek kontrol yang lainnya. Untuk menampilkan *InputBox* ini maka kita harus menggunakan kode program seperti gambar 2.4.

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
  Dim pesan As String
  Pesan = InputBox("Input Nama Anda","Konfirmasi User")

End Sub
```

Gambar 2.4 Kode Menampilkan *Input Box*

penjelasan kode :

InputBox Memiliki minimal dua parameter, yang pertama parameter label yang akan ditampilkan sebagai penanda apa yang akan diminta kepada *user* atau sebagai keterangan yang digunakan untuk memperjelas apa yang akan dilakukan *user*, dan parameter kedua adalah judul atau *title* dari *form inputbox* disini kita bisa membuat dengan memberikan perintah teks dengan menggunakan tanda petik ganda.

2.2 STM32F103C [2]

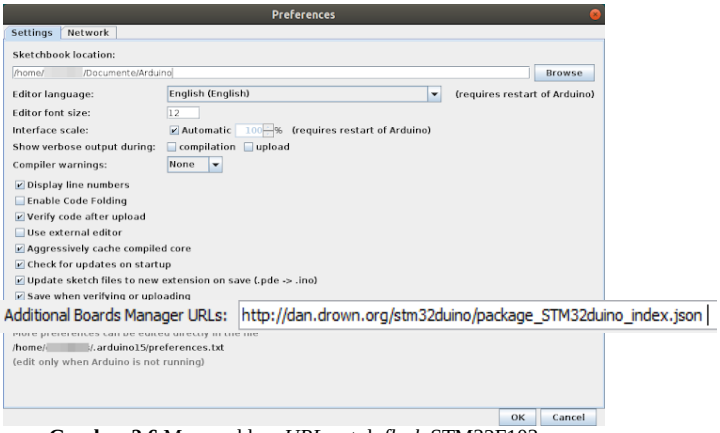
"*Blue Pill*" adalah sebuah papan pengembangan berbasis STM32F103. STM32F103 adalah sebuah alat dengan *Cortex-M3 ARM CPU* yang berkecepatan 72MHz, mempunyai *RAM 20 Kb* dan mempunyai flash memori sebesar 64 atau 128 Kb. *Mikrokontroler* nya mempunyai *port USB*, dua *port serial*, *pin PWM 16 bit*, dan *pin ADC 12 bit.* STM32F103 bekerja pada tegangan 3,3 *V* tetapi beberapa mempunyai pin toleransi 5*V*.

Pemrograman pada *board* ini dapat dipermudah yaitu dengan menggunakan *software Arduino IDE*. Tetapi sebelum itu, sebuah *bootloader Arduino* harus di *flash* kedalam *board* STM32F103. Ini bisa dilakukan melalui *port serial* atau menggunakan *interface debug* pada *MCU* dengan *ST-Link tool.*

Gambar 2.5 STM32F103C

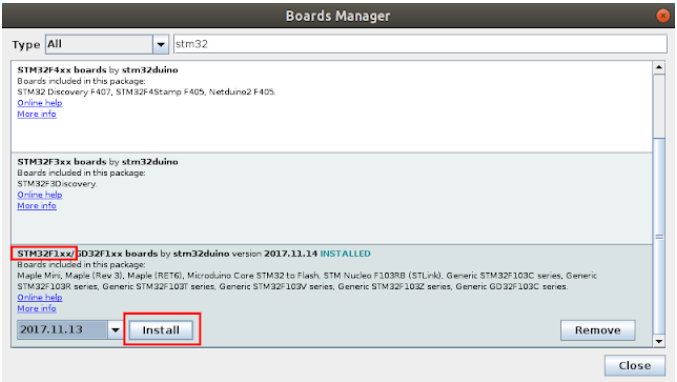
Arduino IDE tidak hadir dengan dukungan *default* untuk papan STM32F103. Oleh karena itu harus di *setting* terlebih dahulu agar papan STM32F103 bisa dientegrasikan dengan *Arduino IDE*. Berikut *step* untuk semua sistem operasi :

Jalankan *Arduino IDE*, pilih *File – Preferences*, dan tambahkan *URL* ini pada *board manager* yang tersedia " ***http://dan.drown.org/stm32duino/package_STM32duino_index.json*** " klik *OK* untuk menutup dialog. Dapat dilihat pada gambar 2.6.



Gambar 2.6 Memasukkan *URL* untuk *flash* STM32F103

Setelah itu, pilih *Tools – Board Manager*. Didalam dialog yang muncul, cari *STM32* dan pilih *STM32F1XX* untuk "*Blue Pill*". Jika *board* menggunakan *chipset* yang berbeda (contoh : STM32F3) pilih seperti yang tertera pada *board* lalu *install*. Untuk *step* nya bisa dilihat pada gambar 2.7.



Gambar 2.7 Meng*instal board* STM32F103 ke *Arduino IDE*

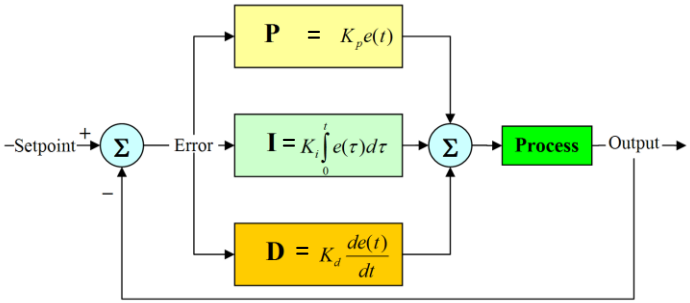
Sebelum benar – benar mengunggah *sketches* ke dalam *board*, dibutuhkan *bootloader STM32duino*. Disini menggunakan sebuah *ST – Link V2 clone* untuk memasukkan *bootloader* nya. Koneksi ke *board* STM32F103 sangat mudah, menggunakan *SWCLK*, *SWDIO*, *GND* dan *pin 3,3V*. sambungkan semua menggunakan kabel *jumper* yang sudah tersedia pada *ST – Link clone*. Pindahkan *jumper* untuk *set BOOT0 to 1*.

2.3 *Proporsional Integral Derivative (PID)* [3]

Didalam suatu sistem kontrol kita mengenal adanya beberapa macam aksi kontrol, diantaranya yaitu aksi kontrol proporsional, aksi kontrol integral dan aksi kontrol derivative. Masing-masing aksi kontrol ini mempunyai keunggulan-keunggulan tertentu, dimana aksi kontrol proporsional mempunyai keunggulan *rise time* yang cepat, aksi kontrol integral mempunyai keunggulan untuk memperkecil *error* ,dan aksi kontrol *derivative* mempunyai keunggulan untuk memperkecil *error* atau meredam *overshot/undershot*. Untuk itu agar kita dapat menghasilkan *output* dengan *risetime* yang cepat

dan *error* yang kecil kita dapat menggabungkan ketiga aksi kontrol ini menjadi aksi kontrol PID.

Parameter pengontrol Proporsional Integral *derivative* (PID) selalu didasari atas tinjauan terhadap karakteristik yang diatur (*plant*). Dengan demikian bagaimanapun rumitnya suatu *plant*, prilaku *plant* tersebut harus diketahui terlabih dahulu sebelum pencarian parameter PID itu dilakukan.



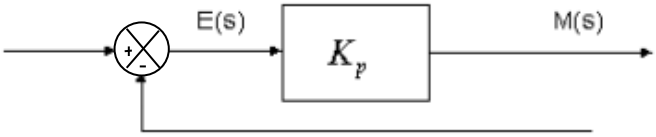
Gambar 2.8 Diagram Blok Kontroler *PID*

Pengontrol proposional memiliki keluaran yang sebanding atau proposional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya). Secara lebih sederhana dapat dikatakan bahwa keluaran pengontrol proporsional merupakan perkalian antara konstanta proposional dengan masukannya. Perubahan pada sinyal masukan akan segera menyebabkan sistem secara langsung mengeluarkan *output* sinyal sebesar konstanta pengalinya.

Gambar 2.8 menunjukkan blok diagram yang menggambarkan hubungan antara besaran *setting*, besaran aktual dengan besaran keluaran pengontrol proporsional. Sinyal keasalahan (*error*) merupakan selisih antara besaran *setting* dengan besaran aktualnya. Selisih ini akan mempengaruhi pengontrol, untuk mengeluarkan sinyal positif (mempercepat pencapaian harga *setting*) atau negatif (memperlambat tercapainya harga yang diinginkan).

Ciri-ciri pengontrol proposional harus diperhatikan ketika pengontrol tersebut diterapkan pada suatu sistem. Secara eksperimen, pengguna pengontrol propoisional harus memperhatikan ketentuan-ketentuan berikut ini :

1. kalau nilai Kp kecil, pengontrol proposional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sisitem yang lambat.
2. kalau nilai Kp dinaikan, respon sistem menunjukan semakin cepat mencapai set point dan keadaan stabil.
3. namun jika nilai Kp diperbesar sehingga mencapai harga yang berlebiahan, akan mengakibatkan sistem bekerja tidak stabil, atau respon sistem akan berosolasi.



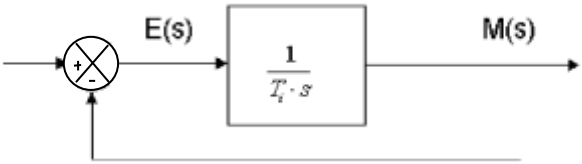
Gambar 2.9 Diagram Blok Kontroler *P*

Pengontrol integral berfungsi menghasilkan respon sistem yang memiliki kesalahan keadaan stabil nol. Jika sebuah *plant* tidak memiliki unsur *integrator (1/s)*, pengontrol proposional tidak akan mampu menjamin keluaran sistem dengan kesalahan keadaan stabilnya nol. Dengan pengontrol integral, respon sistem dapat diperbaiki, yaitu mempunyai kesalahan keadaan stabilnya nol.

Pengontrol integral memiliki karaktiristik seperti halnya sebuah integral. Keluaran sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan. Keluaran pengontrol ini merupakan penjumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan.

Sinyal keluaran pengontrol integral merupakan luas bidang yang dibentuk oleh kurva kesalahan penggerak. Sinyal keluaran akan berharga sama dengan harga sebelumnya ketika sinyal kesalahan berharga nol. Gambar 2.10 menunjukkan blok

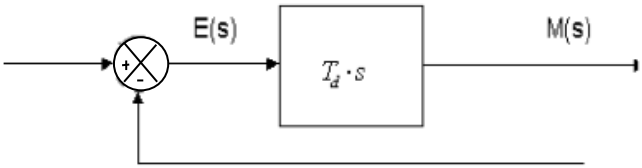
diagram antara besaran kesalahan dengan keluaran suatu pengontrol integral.



Gambar 2.10 Diagram Block Kontroler *I*

Ketika digunakan, pengontrol integral mempunyai beberapa karakteristik berikut ini:

1. keluaran pengontrol membutuhkan selang waktu tertentu, sehingga pengontrol integral cenderung memperlambat respon.
2. ketika sinyal kesalahan berharga nol, keluaran pengontrol akan bertahan pada nilai sebelumnya.
3. jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukkan kenaikan atau penurunan yang dipengaruhi oleh besarnya sinyal kesalahan dan nilai Ki.
4. konstanta integral Ki yang berharga besar akan mempercepat hilangnya offset. Tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengontrol.

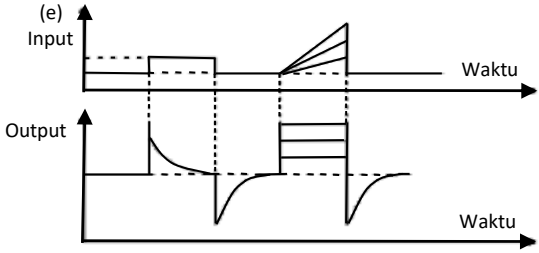


Gambar 2.11 Diagram Blok Kontroler *D*

Keluaran pengontrol *Derivative* memiliki sifat seperti halnya suatu operasi differensial. Perubahan yang mendadak pada masukan pengontrol, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.11 menunjukkan blok diagram

yang menggambarkan hubungan antara sinyal kesalahan dengan keluaran pengontrol.

Gambar 2.12 menyatakan hubungan antara sinyal masukan dengan sinyal keluaran pengontrol *Derivative*. Ketika masukannya tidak mengalami perubahan, keluaran pengontrol juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk *impuls*. Jika sinyal masukan berubah naik secara perlahan (fungsi *ramp*), keluarannya justru merupakan fungsi *step* yang besar magnitudnya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan faktor konstanta diferensialnya.



Gambar 2.12 Kurva waktu hubungan *input-output* pengontrol *Derivative*

Karakteristik pengontrol *derivative* adalah sebagai berikut:

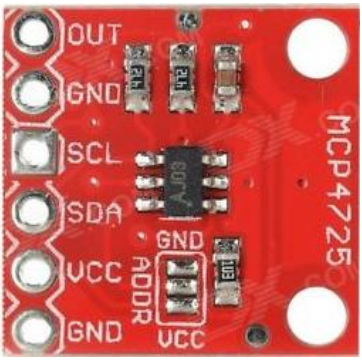
1. pengontrol ini tidak dapat menghasilkan keluaran bila tidak ada perubahan pada masukannya (berupa sinyal kesalahan).
2. jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengontrol tergantung pada nilai Td dan laju perubahan sinyal kesalahan. (Powel, 1994, 184).
3. pengontrol *derivative* mempunyai suatu karakter untuk mendahului, sehingga pengontrol ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi pengontrol *derivative* dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat

korektif, dan cenderung meningkatkan stabilitas sistem .

2.4 *Digital to Analog Converter* MCP4725 (*DAC*) [4]

DAC adalah perangkat yang digunakan untuk mengkonversi sinyal masukan dalam bentuk *digital* menjadi sinyal keluaran dalam bentuk *analog* (tegangan).Tegangan keluaran yang dihasilkan *DAC* sebanding dengan nilai *digital* yang masuk ke dalam *DAC*.Sebuah *DAC* menerima informasi *digital* dan mentransformasikannya ke dalam bentuk suatu tegangan *analog*. Informasi *digital* adalah dalam bentuk angka *biner* dengan jumlah digit yang pasti. Konverter D/A dapat mengonversi sebuah *word digital* ke dalam sebuah tegangan *analog* dengan memberikan skala *output analog* berharga nol ketika semua bit adalah nol dan sejumlah nilai maksimum ketika semua bit adalah satu. Angka *biner* sebagai angka pecahan. Aplikasi *DAC* banyak digunakan sebagai rangkaian pengendali (*driver*) yang membutuhkan *input analog* seperti *motor AC* maupun *DC*, tingkat kecerahan pada lampu, pemanas (*Heate*r) dan sebagainya.

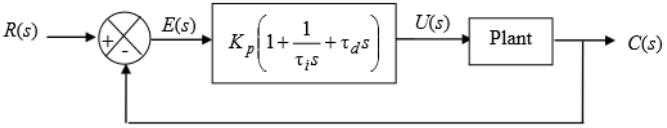
Umumnya DAC digunakan untuk mengendalikan peralatan komputer. Untuk aplikasi modern hampir semua DAC berupa rangkaian terintegrasi (*IC*), yang diperlihatkan sebagai kotak hitam memiliki karakteristik *input* dan *output* tertentu. *SDA* adalah *pin serial* data dari *interface I2C*. *pin SDA* digunakan untuk menulis atau membaca *register DAC* dan data *EEPROM*. *pin SDA* adalah sebuah *open-drain N-chanel driver*. oleh karena itu, dibutuhkan sebuah *pull-up resistor* dari jalur *VDD* ke *pin SDA*. kecuali untuk kondisi *start* dan *stop*. data di *pin SDA* harus stabil selama *clock* dalam keadaan *HIGH*. keadaan *HIGH* atau *LOW* dari *pin SDA* hanya dapat berubah ketika sinyal *clock* di *pin SCL* bernilai *LOW*. *SCL* adalah *pin serial clock* pada *interface I2C*. MCP4725 hanya bertindak sebagai *slave* dan *pin SCL* hanya menerima *serial clock* dari luar. data *input* dari *master device* digeser ke *pin SDA* di *rising edges* pada *SCL clock* dan keluaran dari MCP4725 terjadi saat *falling edges* pada *SCL clock*. *pin SCL* adalah sebuah *open-drain N-chanel driver*. oleh karena itu, dibutuhkan *pull-up resistor* dari jalur *VDD* ke *pin SCL*.



Gambar 2.13 Modul *DAC* MCP4725

2.5 Desain Kontrol *PID Ziegler – Nichols* [5]

Suatu kontroler *PID* yang diterapkan pada suatu *plant* akan menghasilkan sistem umpan balik yang hubungan antara masukan dan keluaran dinyatakan dalam diagram blok pada gambar 2.14.



Gambar 2.14 Diagram blok PID

Dimana fungsi alih kontroler *PID* adalah:

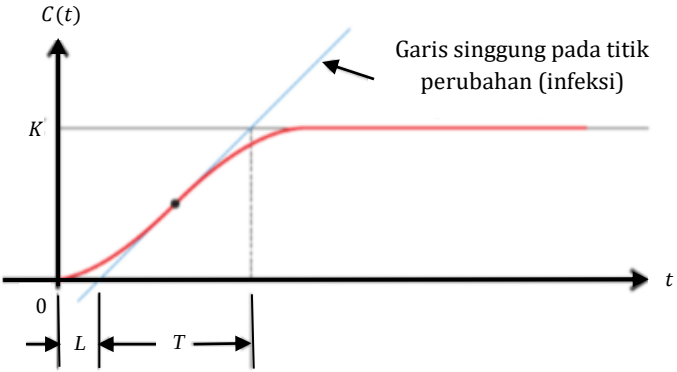
$$\frac{U(s)}{E(s)} = \text{Kp}\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \quad \text{.....(2.1)}$$

Aturan *Zieger & Nichols* menentukan nilai parameter Kp, τi, dan τd berdasarkan pada *rerspon plant* terhadap masukan sinyal *step* secara eksperimental atau berdasarkan pada nilai Kp yang dihasilkan dalam kestabilan marginal bila hanya aksi kendali proporsional yang digunakan. Ada dua metode penalaan *Ziegler – Nichols* yang bertujuan mencapai maksimum *overshoot 25 %*.

2.5.1. Metode pertama aturan *Ziegler - Nichols*

Dalam metode pertama, kita perlu mendapatkan *respon plant* terhadap masukan sinyal *step*. Jika *plant* tidak mengandung

integrator atau kutub pasangan komplek yang dominan, maka kurva *respon step plant* tersebut kelihatan seperti kurva bentuk S.. Jika *respon plant* tidak memiliki kurva berbentuk S, metode ini tidak berlaku. Kurva *respon step* dapat dihasilkan secara eksperimen atau dari simulasi dinamik sistem. Kurva *respon step b*erbentuk S dapat kita lihat seperti gambar 2.15.



Gambar 2.15 Kurva respon bentuk S

Kurva berbentuk S dikarakteristikkan oleh dua parameter yaitu waktu tunda L dan konstanta waktu tunda T. Konstanta waktu tunda T ditentukan dengan menggambarkan garis singgung pada titik perubahan kurva berbentuk S dan menentukan perpotongan garis singgung dengan sumbu waktu dan garis c(t) = K. Fungsi alih *loop* tertutup plant dengan kurva *respon step* berbentuk S ini dapat didekati dengan sistem orde pertama dengan keterlambatan *transport* dengan rumus :

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \quad \text{.....(2.2)}$$

Ziegler dan Nichols menyarankan penentuan nilai Kp, τi, dan τd berdasarkan rumus yang diperlihatkan pada tabel 2.1. Berdasarkan tabel, dapat kita ketahui bahwa parameter Kp, τi, dan τd merupakan fungsi dari waktu tunda L dan konstanta waktu T .

Tipe kontroler	Kp	τ_i	τ_d
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\frac{T}{L}$	2L	0,5L

Tabel 2.1 Rumus Kp, τi, dan τd

Fungsi alih kontroler *PID* berdasarkan metode pertama aturan *Ziegler Nichols* adalah :

$$\frac{U(s)}{E(s)} = Kp\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \quad \text{.....(2.3)}$$

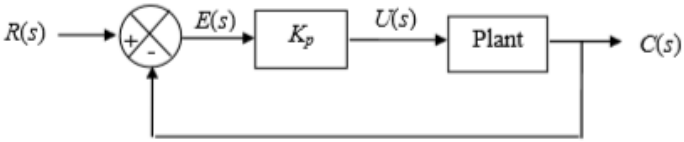
$$= 1{,}2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0{,}5Ls\right)$$

$$\frac{U(s)}{E(s)} = 0{,}6T\,\frac{\left(s + \frac{1}{L}\right)^2}{s}$$

Jadi kontroler *PID* memiliki kutub pada titik asal dan dua nilai nol pada s = -1/L.

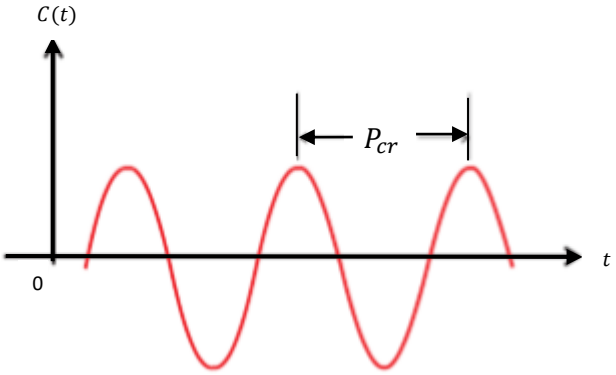
2.5.2 Metode kedua aturan *Ziegler – Nichols*

Dalam metode kedua, mula – mula kita tentukan τi = ∞ dan τd = 0. Dengan menerapkan kontroler *proporsional* pada *plant* seperi diagram blok pada gambar2.16. Kita atur nilai Kp dari nol ke suatu nilai kritis Kcr. Dalam hal ini, mula-mula keluaran *plant* memiliki osilasi yang berkesinambungan dengan periode Pcr seprti gambar 2.17.



Gambar 2.16 Contoh Diagram blok *plant* menggunakan metode kedua aturan *Ziegler – Nichols*

Jika keluaran tidak memiliki osilasi yang berkesinambungan untuk nilai Kp maka metode kedua aturan *Ziegler-Nichols* ini tidak berlaku.



Gambar 2.17 Contoh Osilasi yang terjadi pada gelombang

Ziegler dan Nichols menyarankan penentuan nilai Kp, τi, dan τd berdasarkan rumus yang diperlihatkan pada tabel 2.2.

Tipe kontroler	Kp	τ_i	τ_d
P	0,5Kcr	∞	0
PI	0,9 K_{cr}	$\frac{L}{1,2} P_{cr}$	0
PID	0,6 K_{cr}	0,5 P_{cr}	0,125 P_{cr}

Tabel 2.2 Rumus Kp, τi, dan τd aturan kedua Ziegler – Nichols

Fungsi alih kontroler *PID* berdasarkan metode kedua aturan *Ziegler-Nichols* adalah

$$\frac{U(s)}{E(s)} = Kp\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \text{.....(2.4)}$$

$$= 1{,}2K_{cr}\left(1 + \frac{1}{0{,}5P_{cr}s} + 0{,}125P_{cr}s\right)$$

$$\frac{U(s)}{E(s)} = 0{,}075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

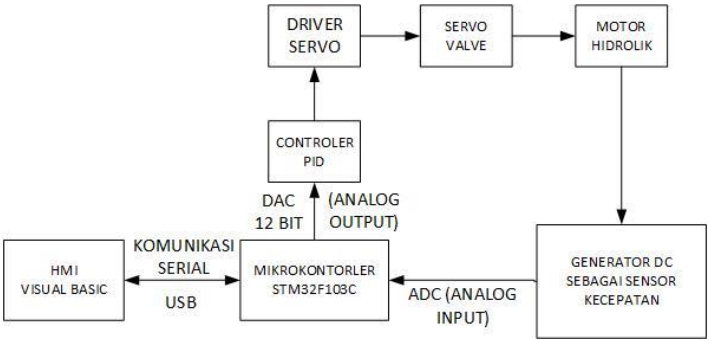
Jadi kontroler *PID* memiliki kutub pada titik asal dan dua nilai nol pada s = -4/Pcr

BAB III
PERENCANAAN DAN IMPLEMENTASI

Pada bab ini akan dibahas mengenai perancangan dan pembuatan sistem dari Tugas Akhir yang akan dikerjakan. Adapun secara garis besar membahas tentang blok diagram sistem, perancangan *hardware* dan perancangan *software.*

3.1 Blok Diagram Sistem

Pada perancangan dan pembuatan Tugas Akhir yang berjudul "Sistem Kontrol dan *HMI* pada modul Elektro Hidrolik" ini akan dijelaskan mengenai cara kerja dari perancangan sistem pemanas yang akan dibuat yakni sebagai berikut :



Gambar 3.1 Blok Diagram fungsi Sistem modul Hidrolik

Diagram blok pada gambar 3.1 menjelaskan bahwa *HMI* yang menggunakan *software visual basic* berperan sebagai pemberi informasi dalam bentuk data dan grafik serta dapat mengkontrol jalannya sistem dengan memberi perintah *start*, *stop* dan memberikan *set point*. Perintah tersebut diterima dan dikirim ke *mikrokontroler* untuk diolah menggunakan komunikasi *serial* melalui *USB*. *Mikrokontroler* yang digunakan adalah STM32F4103C dengan pembacaan *ADC* 12bit. Setelah mengolah perintah maka *mikrokontroler* akan mengeluarkan *output* berupa sinyal *DAC* yang digunakan sebagai *input* pada rangkaian kontroler PID. *Output* dari rangkaian kontroler *PID* digunakan untuk mengatur buka tutup *servo valve* yang akan

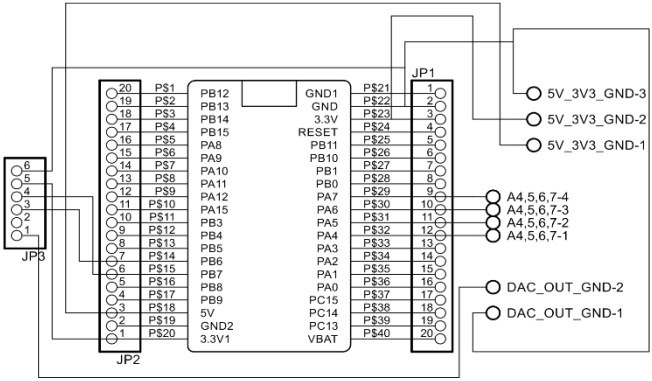
mengatur kecepatan putar dari piringan. Putaran tersebut di baca oleh sensor *tachometer* yang berupa generator *DC* lalu data tersebut dimasukkan ke *mikrokontroler* melalui *ADC*. Setelah mendapatkan semua data yang diperlukan seperti pengukuran dari sensor kecepatan dan *set point*, maka yang dilakukan selanjutnya adalah mencari *transfer function* dari sistem untuk menghitung kp,kd,ki serta diterapkan ke rangkaian kontroler *PID* yang sudah dibuat.

3.2 Perancangan *Hardware*

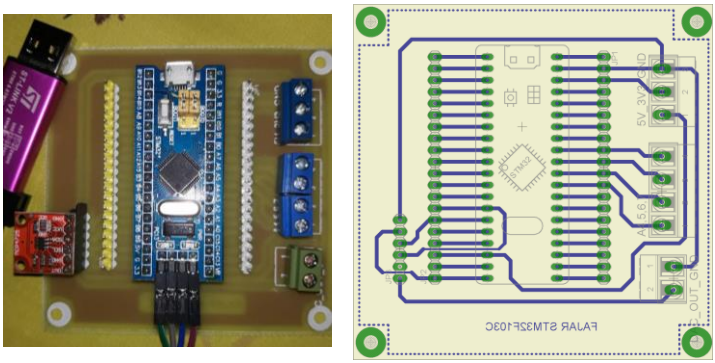
Hardware yang akan dibuat adalah berupa rangkaian – rangkaian yang digunakan untuk mengoptimalkan sistem dalam Tugas Akhir yaitu Rangkaian *Shield* STM32F103C.

3.2.1 Perancangan rangkaian *shield* STM32F103C

Pada tahap ini dibuatlah rangkaian *shield* STM32F103C yang digunakan untuk menggabungkan semua rangkaian *mikrokontroler* STM32F103C dengan semua sensor atau *port* – *port* untuk sensor yang akan digunakan pada modul Hidrolik tersebut. Sensor yang awalnya dihubungkan ke *mikrokontroler* dengan menggunakan kabel dirubah menjadi satu dengan *bord* STM32F103C karena jika sambungan kabel dari sensor ke *mikrokontroler* goyang atau renggang maka akan menggangu pengiriman data dari sensor ke *mikrokontroler* yang berakibat pembacaan tidak bisa akurat.



Gambar 3.2 *Schematic* Rangkaian *Shield* STM32F103C



(a) (b)

Gambar 3.3 (a) Rangkaian jadi dan (b) *Board* dari Rangkaian *Shield* STM32F103C

Pada gambar 3.3 dapat dilihat bahwa pada rangkaian *Shield* STM32F103C terdiri dari beberapa *port* sensor. Terdapat 4 *port* sensor yang bisa membaca *digital* maupun *analog* karena menurut *datasheet* 4 *port* tersebut merupakan *port ADC* sekaligus *port input output digital*. Juga terdapat *port* untuk menempatkan modul *DAC* MCP4725 dengan ditambahkan *port* untuk *output DAC* nya.

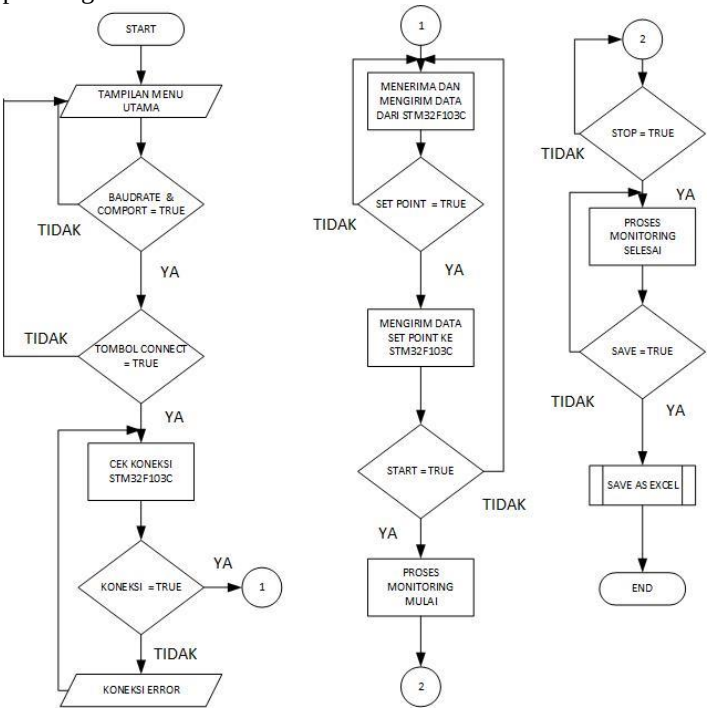
3.3 Perancangan *Software*

Perancangan *software* bertujuan sebagai kontrol sistem secara otomatis dengan menggunakan STM32F103C sehingga sistem dalam pengerjaan Tugas Akhir ini dapat bekerja sesuai dengan yang diinginkan. Setelah merancang berbagai *hardware* yang dibutuhkan, maka selanjutnya dibutuhkanlah perancangan *software* dengan langkah awal yakni membuat perancangan *HMI* agar sistem dapat dijalankan dan di *monitor* melalui *laptop* atau Komputer. Setelah itu membuat perancangan program STM32F103C menggunakan Bahasa c.

3.3.1 Perancangan HMI

Dalam tahap perancangan *HMI* ini di lakukan agar dapat merepresentasikan semua data yang ada dalam STM32F103C untuk memberikan informasi kepada praktikan yang sedang

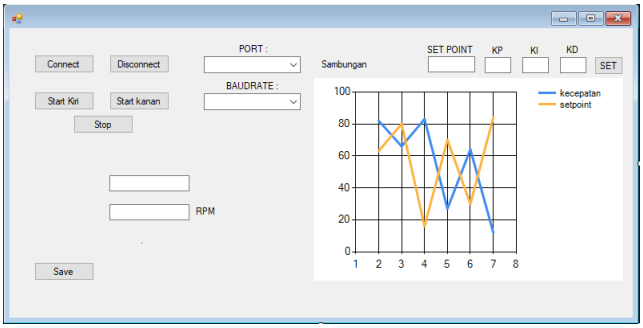
menjalankan modul hidrolik tersebut. *HMI* dibuat menggunakan *software Visual Basic* yang ada didalam *Visual Studio 2015*. Seperti pemrograman pada umumnya, sebelum membuat program pada *Visual Basic* terlebih dahulu harus menyusun diagram penyusunan program yang disebut *flowchart*. *Flowchart* digunakan untuk memudahkan *programmer* untuk meneliti kembali program yang telah dibuat jika terjadi kesalahan dan juga sebagai panduan awal untuk membuat program pada *Visual Basic*. Pada tugas akhir ini telah dibuat *flowchart* seperti gambar 3.4. yang digunakan sebagai panduan untuk memprogram *HMI* pada tugas akhir ini.



Gambar 3.4 *Flowchart* pemrograman *HMI* pada *Visual Basic*

Pada gambar 3.4 dapat dilihat bahwa *HMI* mempunyai tampilan awal yang akan muncul setelah membuka program yang telah dibuat. Pada tampilan awal harus memasukkan atau

memilih *baudrate* dan *Comport* yang akan digunakan untuk komunikasi *serial*. Program *HMI* tidak akan dilanjutkan sebelum keadaan *baudrate* dan *comport* terpenuhi. Lalu setelah memilih *baudrate* dan *Comport* tekan tombol *connect* yang akan mengkoneksikan antara *HMI* dengan STM32F103C dan program akan memeriksa koneksi antar keduanya. Jika *error* maka akan di beri perintah untuk merubah *Comport* atau *Baudrate* dan jika berhasil maka program akan dilanjutkan ke *step* berikutnya yaitu *HMI* berhasil menerima dan mengirim data ke STM32F103C. untuk mengawali penggunaan modul Hidrolik terlebih dahulu mengisi *set point* sesuai yang diharapkan. *Set point* di isi lewat *HMI* dan akan dikirimkan ke STM32F103 melalui komunikasi *serial*. *Set point* harus diisi, jika *set point* tidak diisi maka program tidak akan melanjutkan tugasnya. Setelah mengisi *set point* untuk memulai kinerja modul hanya dengan menekan tombol *start* pada *HMI* dan modul akan bisa berjalan mengirimkan data – data yang diperlukan kehadapan praktikan melalui *HMI* yang telah dibuat. Jika dirasa data yang ditampilkan sudah memenuhi persyaratan sesuai yang diharapkan maka untuk menghentikan proses kinerja modul hidrolik tekan tombol *stop* yang ada pada *HMI* dan seketika proses yang dilakukan oleh modul Hidrolik akan berhenti tetapi data yang telah diambil tetap ada untuk diolah kembali. Setelah berhenti jika ingin mengolah data yang sudah diambil, hanya dengan menekan tombol *save* yang ada pada *HMI* untuk menampilkan data yang telah diambil dalam bentuk *Microsoft Excel* dan dapat disimpan untuk diolah sebagai data praktikum.

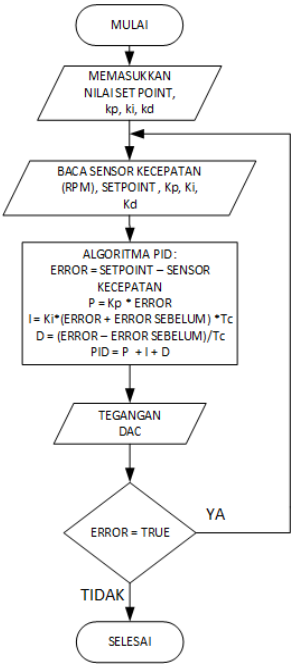


Gambar 3.5 *Screenshot* Tampilan *HMI* pada *Visual Basic*

Untuk gambar 3.5 menunjukkan tampilan dari *form* awal yang telah dibuat untuk tugas akhir ini menggunakan *software Visual Basic*. Pada gambar 3.5 dapat dilihat juga yang ditampilkan pada HMI ini adalah grafik, nilai *set point*, nilai sensor kecepatan, dan data nilai yang dikirimkan lewat *serial* dari STM32F103 sebelum melewati proses *parsing data*.

3.3.2 Perancangan Program STM32F103C

Dalam pembuatan tugas akhir yang berjudul "Sistem kontrol dan *HMI* pada Modul Hidrolik" ini tidak hanya ditunjang oleh perancangan dalam hal *hardware* saja, tetapi juga harus didukung dengan pemrograman *software* yang tepat dan efisien juga. Sebelum mulai membuat program yang akan di integrasikan kedalam sistem haruslah membuat *flowchart* terlebih dahulu untuk memudahkan *programmer* dalam meneliti kesalahan jika terjadi *error* pada program tersebut. Untuk *flowchart* yang sudah dibuat dapat dilihat pada gambar 3.6.



Gambar 3.6 *Flowchart* Program STM32F103C

Dengan melihat gambar *flowchart* yang telah disusun pada gambar 3.6 bisa diketahui bahwa program dimulai dengan memasukkan nilai *set point*, Kp, Ki, dan Kd yang diinginkan. *Set point* ini berisi nilai variabel kecepatan yang diinginkan untuk menjalankan sistem. Setelah itu sistem akan berjalan dan membaca sensor kecepatan untuk bisa mengetahui berapa kecepatan saat dijalankan. Jika data sensor kecepatan sudah didapat, maka langkah selanjutnya yaitu membandingkan dengan *set point*. Jika nilai *rpm* lebih kecil dari nilai *set point* maka pada *flowchart* disebutkan nilai tegangan *output* pada *DAC* di naikkan. Jika sudah memenuhi syarat maka program akan lanjut ke *step* berikutnya. Pada *step* berikutnya jika *rpm* lebih kecil dari *set point* maka tegangan pada *DAC* akan diturunkan sehingga *servo valve* akan perlahan menutup membuat piringan berputar melambat. Jika sudah memenuhi syarat, program akan memutar lagi keatas dan begitu seterusnya.

3.3.3 Perancangan PID

Pada perancangan *PID* ini sangat dibutuhkan untuk mengetahui nilai Kp, Ki, dan Kd yang akan kita gunakan dan implementasikan kedalam modul hidrolik. Dengan menggunakan data yang ada pada bab 4 gambar 4.1 yang merupakan gambar hasil data tanpa kontroler dapat mencari waktu tunda L dan konstanta waktu tunda T dengan metode *Ziegler Nichols* yang referensinya dicantumkan pada bab 2. Jika melihat pada bab 4 gambar 4.1 didapatkan L = 6 dan T= 47. Setelah mengetahui nilai L dan T tindakan selanjutnya adalah mencari Kp, τ_i, τ_d menggunakan rumus yang ada pada tabel 2.1 dan hasilnya adalah:

$$Kp = 1{,}2\ x\ \frac{4{,}7}{6} \quad \text{.....(3.1)}$$
$$= 9{,}4$$

$$\tau_i = 2\ x\ 6 \quad \text{.....(3.2)}$$
$$= 12$$

$$\tau_d = 0{,}5\ x\ 6 \quad \text{.....(3.3)}$$
$$= 3$$

Setelah menemukan nilai Kp τ_i, τ_d langkah selanjutnya mencari Ki dan Kd dengan rumus berikut.

$$Ki = \frac{Kp}{\tau_i} \text{.....(3.4)}$$
$$= \frac{9{,}4}{12}$$
$$= 0{,}7833$$

$$Kd = Kp\ x\ \tau_d \text{.....(3.5)}$$
$$= 9{,}4\ x\ 3$$
$$= 28{,}2$$

Dari perhitungan – perhitungan yang telah dilakukkan didapatkan nilai Kp = 9,4, Ki = 0,7833, dan Kd = 28,2. Nilai tersebut sudah bisa dimasukkkan kedalam program algoritma *PID* yang ada didalam program STM32F103C. untuk hasil dari rumus *PID* yang telah dirancang dapat dilihat pada bab 4.

BAB IV
UJI UKUR DAN UJI COBA

Pengujian dan analisa pada pengerjaan Tugas Akhir ini dilakukan untuk mengetahui parameter – parameter data yang dibutuhkan dari tujuan perancangan *hardware* dan *software* yang telah dibuat sebelumnya, meliputi :

1. Pengujian Rangkaian modul *DAC* MCP4725.
2. Pengujian Sensor kecepatan (*Motor DC* sebagai *Tachometer*).
3. Pengujian kontroler *PID*.
4. Pengujian pembacaan *HMI* di *visual basic*

4.1 Pengujian Rangkaian Modul DAC MCP4725

Tujuan dari pengujian rangkaian modul *DAC* MCP4725 adalah untuk mengetahui apakah *output DAC* yang dihasilkan sesuai dengan perhitungan yang sudah dibuat. Pengujian dilakukan dengan cara memasukkan nilai diskrit kedalam *DAC* lalu ukur dengan *multimeter* berapa besar tegangan yang keluar dari *output* modul *DAC* tersebut. Dengan cara seperti itu bisa dilihat apakah modul *DAC* MCP4725 berfungsi dengan baik atau tidak. Untuk melihat hasil pengujian dapat dilihat beberapa hasil yang sudah diambil dan dimasukkan dalam tabel 4.1

Tabel 4.1 Data Hasil Pengujian Modul *DAC* MCP4725

NO	Nilai Diskrit	*Vout DAC*	*Vout DAC* (Teori)
1	1	1,1 mv	0.8 mv
2	2	1,9 mv	1,61 mv
3	3	2,8 mv	2,41 mv
4	4	3,7 mv	3.22 mv
5	5	4,5 mv	4,02 mv
6	6	5,4 mv	4,83 mv
7	7	6,3 mv	5,63 mv
8	8	7,2 mv	6,44 mv
9	9	8,0 mv	7,24 mv
10	10	8,9 mv	8,05 mv

11	50	43,7 mv	40,25 mv
12	100	86,6 mv	80,5 mv
13	500	0,41 v	0,40 v
14	1000	0,81 v	0,80 v
15	2000	1,63 v	1,61 v
16	3000	2,44 v	2,41 v
17	4000	3,25 v	3,22 v
18	4095	3,3 v	3,29 v

Pada tabel 4.1 berisi tentang perbandingan antara data yang sudah diambil dan data yang diperoleh dari teori perhitungan. Rumus yang digunakan adalah rumus konversi dari nilai diskrit menjadi nilai *analog* atau tegangan. Rumus tersebut untuk mengetahui berapa nilai tegangan dalam 1 nilai diskrit. Jadi diperoleh 1 nilai diskrit = 0,805 *mv* dan untuk mencari nilai tegangan dari nilai diskrit yang lain tinggal dikalikan dengan 0,805mv.

Berdasarkan data yang telah diperoleh dapat dilihat bahwa semakin meningkatnya nilai diskrit tegangan *output* pada modul *DAC* MCP4725 juga meningkat. Perbandingan antara nilai tegangan *output* dari pengukuran dan nilai tegangan *output* dari teori rumus dapat dilihat ada perbedaan tetapi hanya sedikit dengan selisih paling besar adalah 0,9 *mv* dan tidak terlalu berpengaruh pada sistem yang telah dibuat.

4.2 Pengujian Sensor kecepatan (*Motor DC*)

Pengujian ini ditujukan untuk membandingkan data kecepatan pada pembacaan sensor dengan data kecepatan pembacaan dengan tachometer digital agar mendapatkan hasil yang akurat.

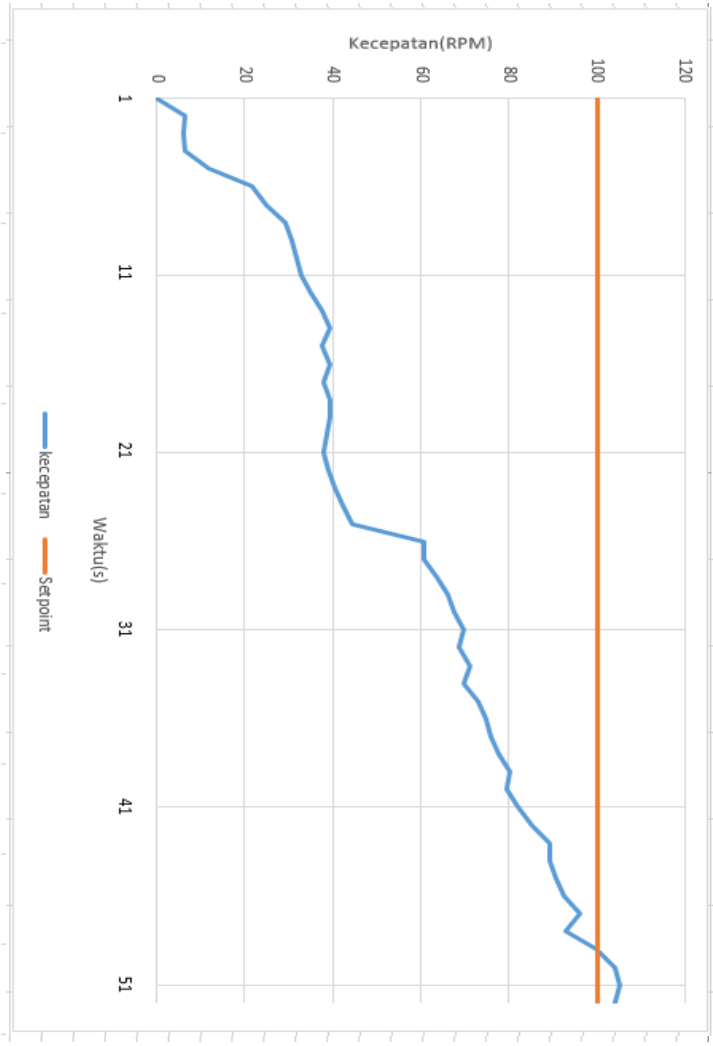
Tabel 4.2 Data Hasil Pengujian *Output* sensor kecepatan

NO	*Vin (V)*	*Vout* Sensor putaran motor *dc (V)*	Kecepatan *(RPM)*
1	0	3.55	175
2	0.02	3.51	170
3	0.67	2.1	100
4	0.88	0.299	78

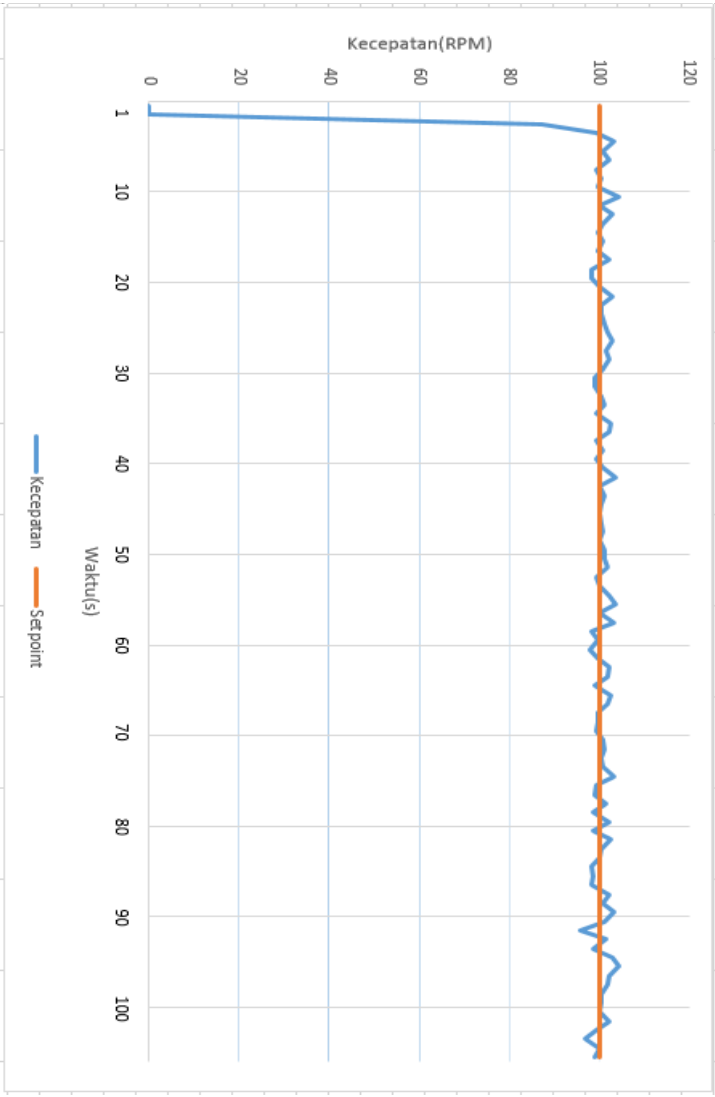
5	1	1.075	55
6	1.2	0.91	48
7	1.5	0.915	48
8	1.6	0.7	10.4
9	1.7	0.65	34
10	1.8	0.4	10.4
11	1.948	0	0
12	2.1	-0.33	26
13	2.2	-0.9	49.7
14	2.3	-1.02	57.9
15	2.4	-1.82	90.2
16	2.5	-2.087	102.5
17	2.6	-2.43	139.2
18	2.7	-2.78	156.2
19	2.8	-3.17	175.2
20	2.9	-3.24	182
21	3.05	-3.44	195
22	3.1	-3.45	197
23	3.19	-3.521	201
24	3.3	-3.521	201

4.3 Pengujian Kontroler *PID*

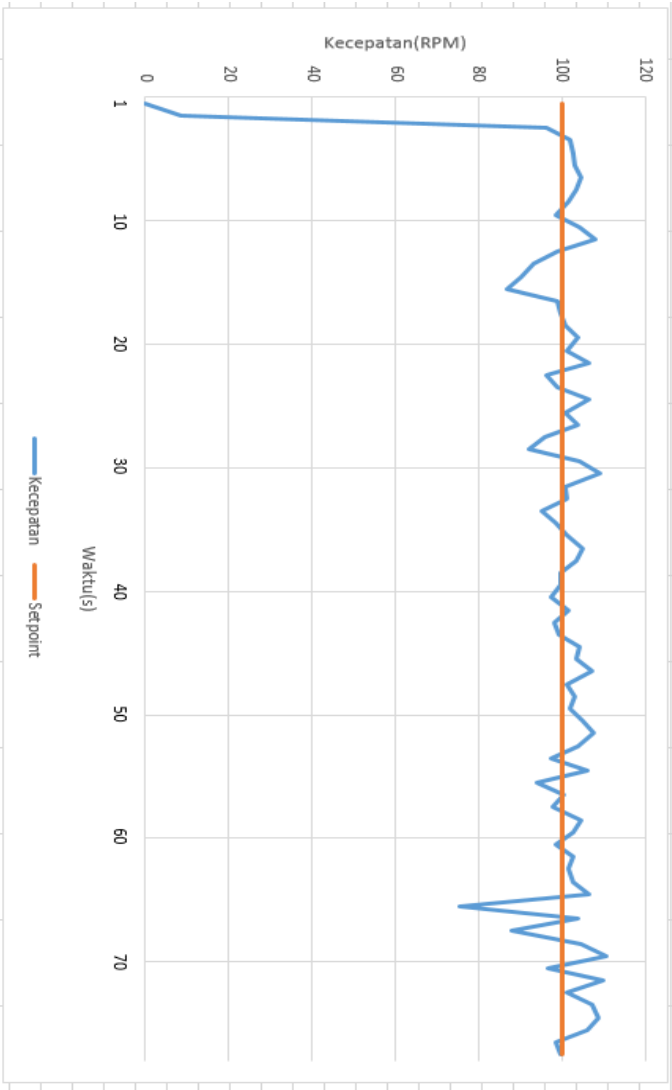
Pengujian ini dilakukan untuk mengetahui kinerja kontroler *PID* pada sistem yang dibuat apakah berpengaruh pada sistem. Pengujian ini dilakukan dengan menggunakan metode *Ziegler Nichols* dan metode coba – coba. Menghasilkan 5 data yang sudah berbentuk grafik pada gambar 4.2 , 4.3, 4.4, 4.5, dan 4.6. Data pada gambar 4.1 diambil dengan menggunakan program *IF* saja pada STM32F103 sehingga dianggap data tanpa kontroler. Menggunakan metode *IF* sama saja dengan hanya menggunakan metode *ON* dan *OFF* karena jika *RPM* lebih dari *setpoint* makan tegangan akan dimatikan dan jika *RPM* kurang dari *setpoint* maka tegangan akan di nyalakan kembali dan begitu seterusnya. Keluaran pada metode *ON* dan *OFF* tidak bisa diatur berapa tegangan yang harus dikurang atau berapa besar nilai tegangan yang harus ditambah jika nilai *RPM* jauh dari *setpoint*. Oleh karena itu yang mengatur pengurangan dan penambahannya adalah kontroler *PID*.



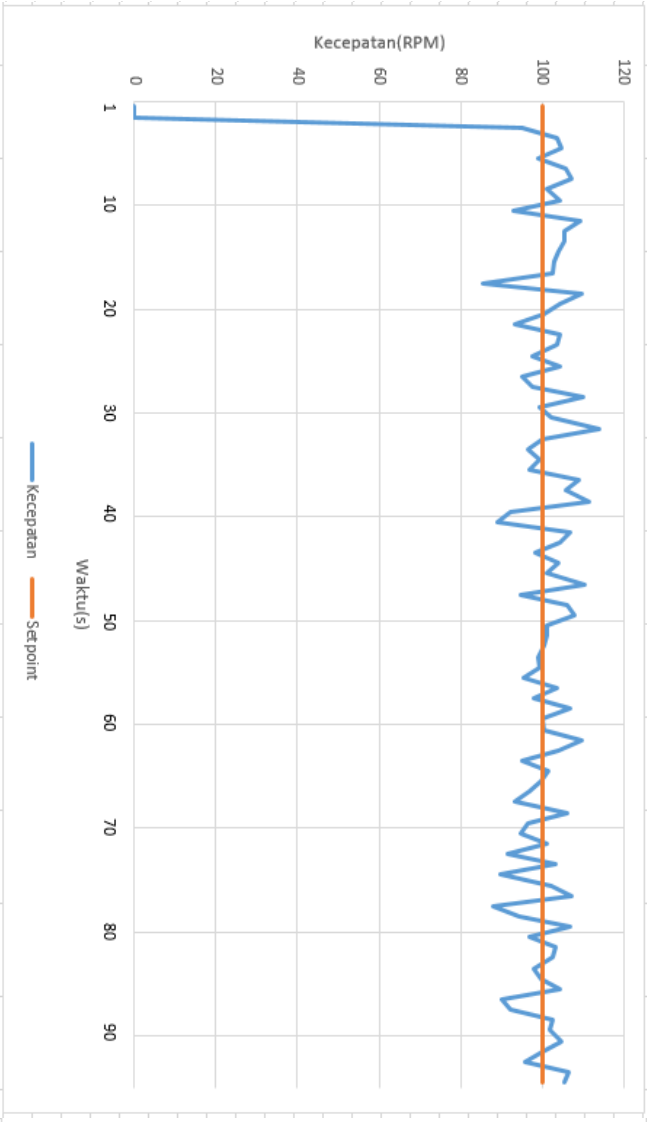
Gambar 4.1 Gambar grafik respon kecepatan tanpa kontroler



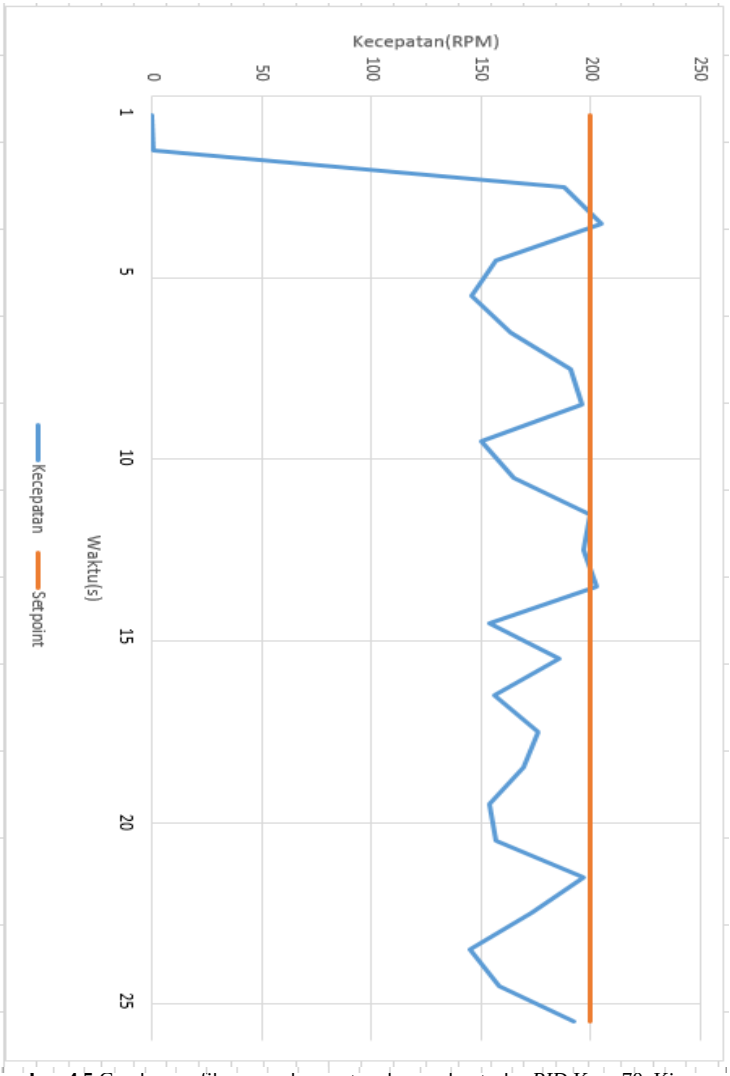
Gambar 4.2 Gambar grafik respon kecepatan dengan kontroler *PID* Kp = 10, Ki = 1, Kd = 28.



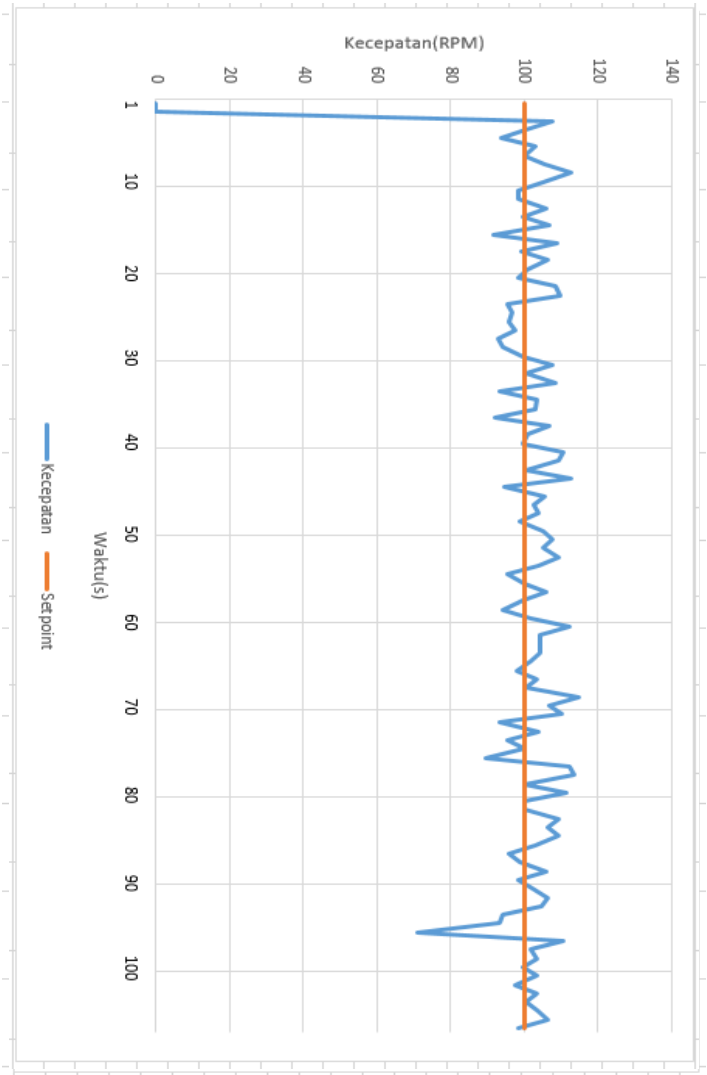
Gambar 4.3 Gambar grafik respon kecepatan dengan kontroler *PID* Kp = 600, Ki = 400, Kd = 1000.



Gambar 4.4 Gambar grafik respon kecepatan dengan kontroler *PID* Kp = 500, Ki = 800, Kd = 800.



Gambar 4.5 Gambar grafik respon kecepatan dengan kontroler *PID* Kp = 70, Ki = 0, Kd = 0.



Gambar 4.6 Gambar grafik respon kecepatan dengan kontroler *PID* Kp = 500, Ki =800, Kd = 1000.

Lima data pengujian yang telah dilakukan menunjukkan bahwa dengan adanya kontroler *PID* sangat berpengaruh terhadap kinerja sistem pada modul Elektro Hidrolik. Terlihat pada gambar 4.1 *plant* sangat lama untuk mencapai *setpoint* dan nilai setelah mencapai *set point* pada 100 *RPM*, nilai kecepatan terus berubah dan tidak tetap pada 100 *RPM*. Oleh karena itu dibutuhkan kontroler agar kinerja sisitem menjadi optimal. Pada gambar 4.2 menunjukkan data hasil dari sistem setelah diberi kontroler *PID* dengan Kp = 10, Ki = 1, dan Kd = 28. Hasilnya waktu untuk plant mencapai *sepoint* lebih cepat daripada tanpa kontroler dan nilai setelah mencapai *setpoint* pada 100 *RPM* sudah *steady* dan hanya berubah sedikit. Kp, Ki ,Kd pada pengujian ini didapat dari metode *Ziegler Nichols* dengan rumus persamaan yang ada pada bab 2 sehingga menghasilkan nilai Kp = 10, Ki = 1, dan Kd = 28.

Pada gambar 4.3, 4.4, 4.5, 4.6 menunjukkan data hasil dari sistem setelah diberi kontroler *PID* yang nilai Kp, Ki, Kd nya ditentukkan dengan metode coba – coba. Dari ke empat data tersebut menunjukkan unutk nilai setelah mencapai *setpoint* masih banyak mengalami osilasi sehingga kurang baik jika diterapkan ke modul Hidrolik langsung karena osilasi yang ditimbulkan bisa merusak *servo valve* yang ada pada modul Elektro Hidrolik tersebut. Tetapi waktu untuk mencapai *setpoint*-nya pada nilai 100 *RPM* sudah lebih cepat dibandingkan data sebelum diberi kontroler.

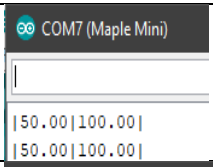
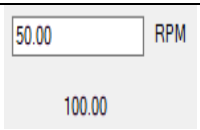
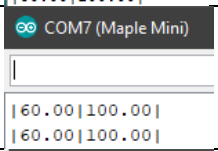
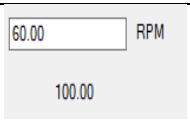
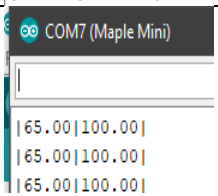
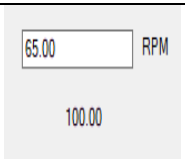
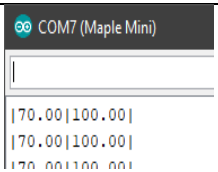
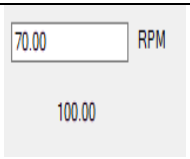
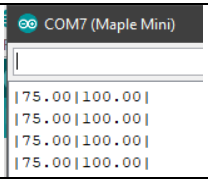
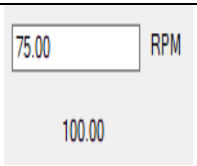
Melihat data setelah melakukkan pengujian didapatkan data yang terbaik untuk diterapkan pada modul Elektro Hidrolik ini adalah data dengan nilai Kp = 10, Ki = 1, Kd = 28. Hasil pengujian data tersebut dapat dilihat pada gambar 4.2.

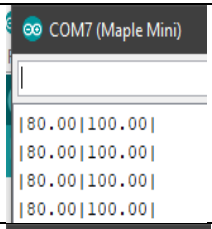
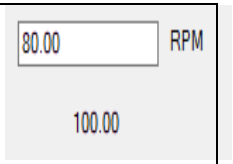
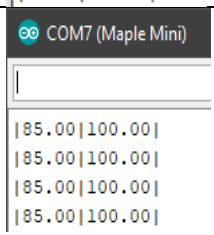
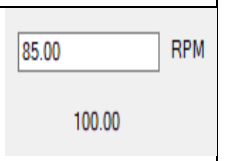
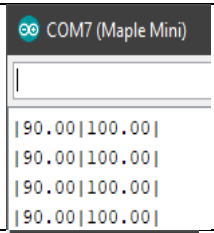
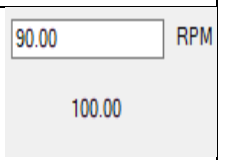
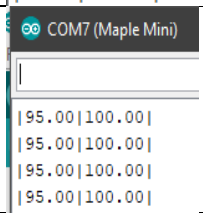
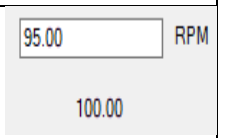
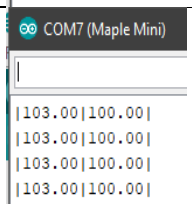
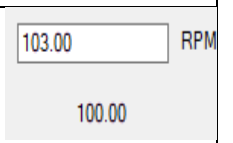
4.4 Pengujian Pembacaan *HMI* di *Visual Basic*

Tujuan dari pengujian pembacaan *HMI* di *visual basic* ini adalah untuk mengetahui apakah data yang dikirimkan dari *mikrokontroler* dan data yang dibaca oleh *HMI* sama atau tidak. Karena jika terjadi perbedaan nilai yang dikirim maka data akan menjadi tidak akurat dan *HMI* tidak bisa dijadikan acuan untuk melakukan pengukuran. Pengujian dilakukan dengan cara

mengirimkan data konstan dari *mikrokontroler* ke *HMI* dan akan terbaca pada kolom *RPM* yang terdapat pada *visual basic*.

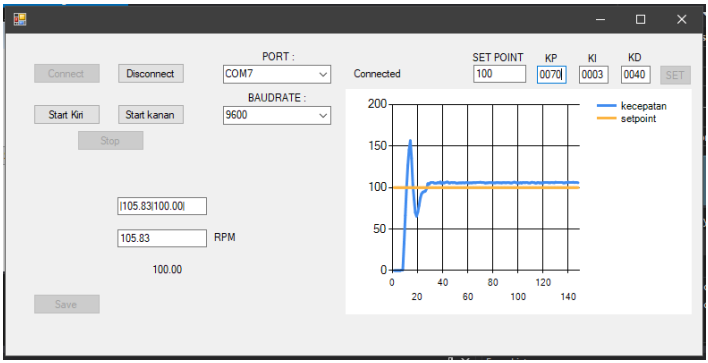
Tabel 4.3 Data Hasil Pengujian Pembacaan *HMI Visual Basic*

NO	Data dari *Mikrokontroler*	Data terbaca di *serial monitor*	Data terbaca di *visual basic*
1	50 *RPM*		
2	60 *RPM*		
3	65 *RPM*		
4	70 *RPM*		
5	75 *RPM*		

6	80 *RPM*		
7	85 *RPM*		
8	90 *RPM*		
9	95 *RPM*		
10	103 *RPM*		

Pada data tabel 4.3 terlihat data yang dikirim dari *mikrokontroler* berupa nilai kecepatan dalam satuan *RPM* yang diperoleh dari sesnsor kecepatan dapat terbaca dengan baik dan

benar pada *HMI* dengan menggunakan *software Visual Basic*. Hasil yang dikeluarkan dari *mikrokontroler* dan hasil dari pembacaan *HMI* selalu menunjukkan nilai yang sama.



Gambar 4.7 *HMI* saat melakukan Proses pengujian pembacaan sensor kecepatan.

Pada gambar 4.7 menunjukkan hasil pengujian saat *HMI* melakukkan proses pembacaan sensor kecepatan saat modul Hidrolik dijalankan. Data hasil *RPM* di tampilkan dalam bentuk angka yang ditempatkan pada kolom *RPM* dan juga dalam bentuk grafik yang dapat dilihat pada gambar 4.7.

----Halaman ini sengaja dikosongkan---

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melakukan tahap perancangan dan pembuatan sistem yang kemudian dilanjutkan dengan tahap pengujian dan analisa maka dapat diambil kesimpulan sebagai berikut:

1. *HMI* yang dibuat pada *software Visual Basic* sudah menunjukkan hasil pembacaan yang *real time* dan sama jika di bandingkan dengan pembacaan di *mikrokontroler*. Jika di *mikrokontroler* membaca 120 *RPM* maka di *HMI* akan muncul nilai yang sama pada kolom pembacaan *RPM* yaitu sebesar 120 *RPM*. Dan semua perubahan data dari nol sampai data yang diinginkan diambil oleh *HMI* dari *mikrokontroler* dan direpresentasikan menggunakan grafik yang ada pada tampilan *HMI* tersebut. Sehingga dapat terlihat dengan jelas perubahan respon yang terjadi saat modul hdrolik dijalankan melalui *HMI*.
2. Kontroler *PID* dapat digunakan untuk mepertahankan nilai kecepatan agar selalu berada pada *setpoint*. Setelah melakukan banyak pengujian dengan Kp = 10, Ki = 1, dan Kd = 28 menghasilkan hasil yang terbaik dimana *setpoint* bisa tercapai lebih cepat dalam waktu hanya 1,5 detik dan nilai kecepatan selalu berada di *setpoint* dengan nilai 100 *RPM* sedangkan jika tidak menggunakan kontroler untuk mencapai *setpoint* dengan nilai 100 *RPM* dibutuhkan waktu 6 detik lamanya. Dari hasil tersebut dapat dikatakan bahwa dengan kontroler *PID rise time* bisa lebih cepat dan nilai kecepatan selalu berada di setpoint sehingga kecepatan motor hidrolik sesuai dengan yang diinginkan.

5.2 Saran

1. Masih banyak kekurangan dari tugas akhir ini seperti sensor yang digunakan hanya satu yaitu sensor kecepatan yang menggunakan *motor DC* sebagai *Tachometer*. Sedangkan sensor yang belum terbaca adalah sensor posisi dan tiga sensor tekanan. Jika ingin

menggunakan modul Hidrolik ini untuk mengambil data posisi dan tekanan disarankan untuk melanjutkan dengan mengambil topik pembacaan sensor posisi dan tekanan pada modul hidrolik ini.

2. Untuk menjalankan modul Hidrolik ini disarankan untuk mencari literatur tentang modul hidrolik ini sebanyak – banyaknya. Karena dalam merancang tugas akhir ini penulis kesulitan menemukan literatur seperti *datasheet* sensor, *datasheet servo valve*, dll. Sehingga untuk menjalankan modul tersebut butuh waktu yang sedikit lama.
3. Untuk menjalankan *software HMI* disarankan untuk menggunakan Komputer atau *laptop* dengan spesifikasi yang tinggi jika ingin tidak mengalami *hang* saat menjalankan modul Hidrolik atau saat pengambilan data berlangsung karena kecepatan proses dari *processor* pada komputer atau *laptop* berpengaruh terhadap kinerja *HMI*.

DAFTAR PUSTAKA

[1] Rolly Yesputra (2017), *"Belajar Visual Basic.Net Dengan Visual studio 2010"*. Kisaran : Royal Asahan Press.

[2] Ddrown. 2009. http://www.stm32duino.com/viewtopic.php?t=844 (Diakses pada 16 april, 2018)

[3] Nurlita Gamayanti, ST (2011), *"Desain Kontroler Proposional Ditambah Integral Ditambah Differensial (PID)"*. Surabaya : Institute Teknologi Sepuluh Nopember.

[4] Datasheet modul DAC MCP4725. https://www.sparkfun.com/datasheets/BreakoutBoards/MCP4725.pdf . (Diakses pada 13 juni, 2018)

[5] Nurlita Gamayanti, ST (2011), *"Desain Kontroler PID Ziegler Nichols"*. Surabaya : Institute Teknologi Sepuluh Nopember.

[6] Nilsson, Johan (2010) *"Position Control of an Electro-Hydraulic Servo-Valve"*. Stockholm Sweden : KTH Royal Institute Of Technology School Of Electrical Engineering.

[7] Johnson, Curtis D (2000), "*Process Control Instumentation Technology sixth edition*". Prentice, US : Upper Saddle River, Prentice-Hall.

[8] BPPT (2012) *"Positioning Control Of The Hydraulic Servo System Performance Analysis"*. Jakarta : BPPT

“Halaman Ini Sengaja Dikosongkan”

LAMPIRAN A
LISTING PROGRAM

A. *Listing* Program STM32F103

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
uint32_t dac_value;
float p;
float i;
float d;
float pid;
float pid1;
float i1;
float i2;
float d1;
float d2;
float nilai1;
float nilai2;
float nilai3;
float nilai4;
char c;
float kp;
float ki;
float kd;
float nilaipos1;
float nilaipos;
float input;
float Setpoint;
float error;
float last_error;
float last_errord;
float tc = 0.001;
String data1;
String data2;
String data3;
String data4;
String packet;
void parsePacket(String myPacket);
```

```
Adafruit_MCP4725 dac;
void setup() {
  pinMode(PA5, INPUT);
  //digitalWrite(PA5, LOW);
  //pinMode(PA7,INPUT);
  //digitalWrite(PA7, LOW);
  dac.begin(0x62);
  dac_value = 2435;
  dac.setVoltage(dac_value, false);
  //  Setpoint = 200;
  //  kp = 20;
  //  ki = 1;
  //  kd = 4;
  Serial.begin(9600);

}

void loop() {
  baser();
  switch (c) {
    case 'a':
      // digitalWrite(PA5, HIGH);
      nilaipos = analogRead(PA4);
      nilaipos1 = nilaipos * 0.00080;
      input = nilaipos1 * 108.79;
      error = Setpoint - input ;
      p = kp * error;
      i1 = error + last_error;
      i2 = i1;
      i = ki * i2 * tc;
      last_error = error;
      d1 = kd;
      d2 = error - last_errord;
      d = (d1 * d2) / tc;
      last_errord = error;
      pid = p + d + i;
      //last_error = error;
      if (pid <= 1) {
        pid = 0;
```

```
  }
  pid1 = map(pid, 0, 4095, 2435, 4095);
  if (pid1 >= 4095) {
    pid1 = 4095;
  }
  dac_value = pid1;
  //dac_value = 2435;
  dac.setVoltage(dac_value, false);
  Serial.print("|");
  Serial.print(input);
  Serial.print("|");
  //Serial.print("input = ");
  Serial.print(Setpoint);
  Serial.println("|");
  delay(1);
  //Serial.print("output = ");
  //Serial.println(dac_value);
  break;
case 'b':
  dac_value = 2435;
  dac.setVoltage(dac_value, false);
  system("cls");
  nvic_sys_reset();
  break;
case 'c':
  //digitalWrite(PA7, HIGH);
  nilaipos = analogRead(PA5);
  nilaipos1 = nilaipos * 0.00080;
  input = nilaipos1 * 108.79;
  error = Setpoint - input ;
  p = kp * error;
  i1 = error + last_error;
  i2 = i1;
  i = ki * i2 * tc;
  last_error = error;
  d1 = kd;
  d2 = error - last_errord;
  d = (d1 * d2) / tc;
  last_errord = error;
```

```
      pid = p + d + i;
      if (pid <= 1) {
        pid = 0;
      }
      pid1 = map(pid, 0, 4095, 2435, 0);
      if (pid1 >= 4095) {
        pid1 = 4095;
      }
      dac_value = pid1;
      dac.setVoltage(dac_value, false);
      Serial.print("|");
      Serial.print(input);
      Serial.print("|");
      //Serial.print("input = ");
      Serial.print(Setpoint);
      Serial.println("|");
      delay(1);
      //Serial.print("output = ");
      //Serial.println(dac_value);
      break;
    }
  }
  void baser() {
    if (Serial.available())
    {
      c = Serial.read();

      /*Is it end of packet??*/
      if (c == '#')
      {
        //Serial.println(packet);
        parsePacket(packet);
      }
      /*Buffering byte into one packet*/
      else
      {
        packet += c;
        //Serial.println(c);
      }
```

```
  }
}
void parsePacket(String myPacket)
{
  data1 = myPacket.substring(0, 3);
  data2 = myPacket.substring(3, 7);
  data3 = myPacket.substring(7, 11);
  data4 = myPacket.substring(11, 15);
  nilai1 = data1.toInt();
  nilai2 = data2.toInt();
  nilai3 = data3.toInt();
  nilai4 = data4.toInt();
  Setpoint = nilai1;
  kp = nilai2;
  ki = nilai3;
  kd = nilai4;
  //Serial.print("setpoint :");
  Serial.print(Setpoint);
  //Serial.print("kp :");
  Serial.print(kp);
  //Serial.print("ki :");
  Serial.print(ki);
  //Serial.print("kd :");
  Serial.println(kd);
}
```

B. Program utama *HMI* di *Visual basic*.

```
Imports System.IO.Ports
Imports System.Threading.Thread
Imports Microsoft.Office.Interop.Excel
Imports
System.Windows.Forms.DataVisualization.Charting
Public Class Form1
    Dim WithEvents COMPort As New SerialPort
    Dim RXArray(2047) As Char
    Dim RXCnt As Integer
```

```
    Dim Now As String
    Dim engChart As New
System.Windows.Forms.DataVisualization.Charting.Se
ries
    Dim SReader As String


    Private readBuffer As String = String.Empty
    Dim strinput As String
    Private Bytenumber As Integer
    Private ByteToRead As Integer
    Private byteEnd(2) As Char
    Dim prosesoff As Boolean = False
    Dim disconnect As Boolean = False
    Dim data(5) As String
    Dim d1() As String
    Dim d2() As String
    Dim d3() As String
    Dim datar1 As Double
    Dim datar2 As Double
    Dim timerCounter As Integer = 0
    Dim datasetpoint As String
    Dim datasetpoint1 As String
    Delegate Sub SetTextCallback(ByVal [text] As
String)

    Private Sub Btconnect_Click(sender As Object,
e As EventArgs) Handles Btconnect.Click
        If COMPort.IsOpen Then
            COMPort.RtsEnable = False
            COMPort.DtrEnable = False
            COMPort.Close()
            Sleep(200)
        End If
        COMPort.PortName = CBport.Text
        COMPort.BaudRate = CBbaud.Text
        COMPort.WriteTimeout = 2000

        Try
            COMPort.Open()
```

```
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try

        If COMPort.IsOpen Then
            COMPort.RtsEnable = True
            COMPort.DtrEnable = True
            Lbkoneksi.Text = "Connected"
            Btconnect.Enabled = False
            Btdisconnect.Enabled = True
            Btmulai.Enabled = False
            Btstop.Enabled = False
            Btsetpoint.Enabled = True
            btsave.Enabled = False
            Btkiri.Enabled = False
        Else
            Btconnect.Enabled = True
            Btmulai.Enabled = True


        End If
    End Sub

    Private Sub Btdisconnect_Click(sender As
Object, e As EventArgs) Handles Btdisconnect.Click
        If COMPort.IsOpen Then
            COMPort.Close()
            Lbkoneksi.Text = "Disconnected"
            Btconnect.Enabled = True
            Btdisconnect.Enabled = False
        Else
            Btconnect.Enabled = True
            Btdisconnect.Enabled = False
        End If

    End Sub

    Private Sub Btmulai_Click(sender As Object, e
As EventArgs) Handles Btmulai.Click
        If COMPort.IsOpen Then
```

```
            COMPort.Write("a")
            Btstop.Enabled = True
            Btmulai.Enabled = False
            Btkiri.Enabled = False
            btsave.Enabled = False
        Else
            Btmulai.Enabled = False
        End If

    End Sub

    Private Sub Btstop_Click(sender As Object, e
As EventArgs) Handles Btstop.Click
        If COMPort.IsOpen Then
            COMPort.Write("b")
            Timer1.Enabled = False
            Btstop.Enabled = False
            btsave.Enabled = True
        Else
            Btstop.Enabled = False
        End If
    End Sub

    Private Sub CBport_SelectedIndexChanged(sender
As Object, e As EventArgs) Handles
CBport.SelectedIndexChanged
        If COMPort.IsOpen = False Then
            COMPort.PortName = CBport.Text
'pop a message box to user if he is changing ports
        Else
'without disconnecting first.
            MsgBox("Valid only if port is Closed",
vbCritical)
        End If
        'If COMPort.Close Then
        '    Btconnect.Enabled = True
        '    Btdisconnect.Enabled = False
        '    Btkiri.Enabled = False
        '    Btmulai.Enabled = False
        '    btsave.Enabled = False
```

```
        '    Btsetpoint.Enabled = False
        '    Btstop.Enabled = False

        'End If
    End Sub

    Private Sub CBbaud_SelectedIndexChanged(sender
As Object, e As EventArgs) Handles
CBbaud.SelectedIndexChanged
        If COMPort.IsOpen = False Then
            COMPort.BaudRate = CBbaud.Text
'pop a message box to user if he is changing baud
rate
        Else
'without disconnecting first.
            MsgBox("Valid only if port is Closed",
vbCritical)
        End If
    End Sub

    Private Sub Btsetpoint_Click(sender As Object,
e As EventArgs) Handles Btsetpoint.Click
        datasetpoint = TBsetpoint.Text()
        COMPort.Write(datasetpoint)
        COMPort.Write(Tbkp.Text)
        COMPort.Write(Tbki.Text)
        COMPort.Write(Tbkd.Text)
        COMPort.Write("#")
        Timer1.Enabled = True
        Btsetpoint.Enabled = False
        btsave.Enabled = False
        Btmulai.Enabled = True
        Btkiri.Enabled = True
        ''Tbki.Enabled = False
        ''Tbkd.Enabled = False
        ''Tbkp.Enabled = False

    End Sub
```

```
    Private Sub Form1_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        ReDim d1(1)
        ReDim d2(1)
        ReDim d3(1)
        Btconnect.Enabled = True
        Btdisconnect.Enabled = False
        Btkiri.Enabled = False
        Btmulai.Enabled = False
        btsave.Enabled = False
        Btsetpoint.Enabled = False
        Btstop.Enabled = False
        For Each COMString As String In
My.Computer.Ports.SerialPortNames
            CBport.Items.Add(COMString)
            CBbaud.Items.Add(9600)     'Populate
the cmbBaud Combo box to common baud rates used
            CBbaud.Items.Add(19200)
            CBbaud.Items.Add(38400)
            CBbaud.Items.Add(57600)
            CBbaud.Items.Add(115200)
        Next
    End Sub
    Private Sub Receiver(ByVal sender As Object,
ByVal e As SerialDataReceivedEventArgs) Handles
COMPort.DataReceived

        If COMPort.IsOpen Then
            Try
                '================Default Coding
Data Serial==========================
                byteEnd =
COMPort.NewLine.ToCharArray
                Bytenumber = COMPort.BytesToRead
                readBuffer = COMPort.ReadLine()

                Call proses_fix()
            Catch ex As Exception
                'MsgBox("read " & ex.Message)
            End Try
```

```
        End If
    End Sub

    Private Sub proses_fix()
        strinput = Dataawal.Text
        Dim panjang_data As Integer
        Dim x As Integer
        Dim z As Integer
        panjang_data = Len(strinput)

        Dim i As Integer
        i = 0
        z = 0

        'jika mendeteksi "|" pada string
txt_inputdata dimulai string ke1 dengan panjang
data 1
        For x = 1 To Len(readBuffer$)
'looping untuk data ke dua sampai terakhir pada
txt_inputdata
            If Mid(readBuffer$, x, 1) = "|" Then
                z = z + 1
'jika mendeteksi "|" ke dua dimulai ke x dengan
panjang data 1, jadi x sebagai simbol dari "|"
                data(i) = Mid(readBuffer$, z, x -
z) 'data ke i = integer dari pengambilan data
txt_inputdata dimulai dari z(awalnya 2) sampai ke
x - z
                i = i + 1
'i incr dari 0 ke satu, satu ke dua dst...
                z = x
'z di incrm dengan x karena panjang max = x dan
awal data = z maka untuk mendapatkan data = x-z
            End If
        Next x
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As
EventArgs) Handles Timer1.Tick
        If COMPort.IsOpen Then
```

```
            Now = TimeOfDay
            Dataawal.Text = readBuffer
            Datakecepatan.Text = data(1)
            LBsetpoint.Text = data(2)
            Try

grafik.Series.Item("kecepatan").Points.AddY(data(1
))

grafik.Series.Item("setpoint").Points.AddY(data(2)
)
            Catch ex As Exception
            Finally
            End Try
            If Dataawal.Text <> "" Then
                timerCounter = timerCounter + 1

                d1(timerCounter - 1) = data(1)
                d2(timerCounter - 1) = data(2)
                d3(timerCounter - 1) =
Format(TimeOfDay, "hh:mm:ss")

                ReDim Preserve d1(timerCounter +
1)
                ReDim Preserve d2(timerCounter +
1)
                ReDim Preserve d3(timerCounter +
1)
            End If
        End If
    End Sub

    Private Sub btsave_Click(sender As Object, e
As EventArgs) Handles btsave.Click
        Dim xlApp As Application = New
Application()

        If xlApp Is Nothing Then
            MessageBox.Show("Excel is not properly
installed!!")
```

```
            Return
        End If

        Dim sfd As New SaveFileDialog
        If sfd.ShowDialog() = DialogResult.OK Then
            Dim xlWorkBook As Workbook
            Dim xlWorkSheet As Worksheet
            Dim misValue As Object =
System.Reflection.Missing.Value

            xlWorkBook = xlApp.Workbooks.Add()
            xlWorkSheet = xlWorkBook.Sheets(1)

            Dim row As Long = d1.Length
            Dim arr(row + 1, 3) As Object

            arr(0, 0) = "WAKTU"
            arr(0, 1) = "kecepatan"
            arr(0, 2) = "SET POINT"
            For i As Long = 1 To row
                arr(i, 0) = d3(i - 1)
                arr(i, 1) = d1(i - 1)
                arr(i, 2) = d2(i - 1)
            Next

            Dim sCell As Range =
xlWorkSheet.Cells(1, 1)
            Dim eCell As Range =
xlWorkSheet.Cells(row, 3)
            Dim rng As Range =
xlWorkSheet.Range(sCell, eCell)

            rng.Value2 = arr

            xlWorkBook.SaveAs(sfd.FileName)
            xlWorkBook.Close(True, misValue,
misValue)
            xlApp.Quit()
        End If
```

```
        ReDim d1(1)
        ReDim d2(1)
        ReDim d3(1)
        timerCounter = 0
    End Sub

    Private Sub Btkiri_Click(sender As Object, e
As EventArgs) Handles Btkiri.Click
        COMPort.Write("c")
        Btmulai.Enabled = False
        Btkiri.Enabled = False
        Btstop.Enabled = True
    End Sub

End Class
```

LAMPIRAN B
DATASHEET

A. ***Datasheet DAC MCP4725.***

MICROCHIP

MCP4725

12-Bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6

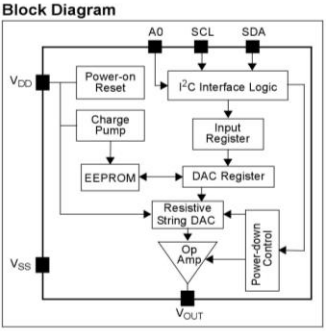
Features

- 12-Bit Resolution
- On-Board Non-Volatile Memory (EEPROM)
- ±0.2 LSB DNL (typical)
- External A0 Address Pin
- Normal or Power-Down Mode
- Fast Settling Time of 6 µs (typical)
- External Voltage Reference (V_{DD})
- Rail-to-Rail Output
- Low Power Consumption
- Single-Supply Operation: 2.7V to 5.5V
- I^2C^{TM} Interface:
  - Eight Available Addresses
  - Standard (100 kbps), Fast (400 kbps), and High-Speed (3.4 Mbps) Modes
- Small 6-lead SOT-23 Package
- Extended Temperature Range: -40°C to +125°C

Applications

- Set Point or Offset Trimming
- Sensor Calibration
- Closed-Loop Servo Control
- Low Power Portable Instrumentation
- PC Peripherals
- Data Acquisition Systems

Block Diagram



DESCRIPTION

The MCP4725 is a low-power, high accuracy, single channel, 12-bit buffered voltage output Digital-to-Analog Convertor (DAC) with non-volatile memory (EEPROM). Its on-board precision output amplifier allows it to achieve rail-to-rail analog output swing.

The DAC input and configuration data can be programmed to the non-volatile memory (EEPROM) by the user using I^2C interface command. The non-volatile memory feature enables the DAC device to hold the DAC input code during power-off time, and the DAC output is available immediately after power-up. This feature is very useful when the DAC device is used as a supporting device for other devices in the network.

The device includes a Power-On-Reset (POR) circuit to ensure reliable power-up and an on-board charge pump for the EEPROM programming voltage. The DAC reference is driven from VDD directly. In power-down mode, the output amplifier can be configured to present a low, medium, or high resistance output load.

The MCP4725 has an external A0 address pin. This A0 pin can be tied to VDD or VSS of the user's application board.

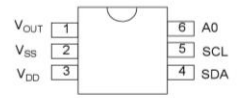
The MCP4725 has a two-wire I^2C™ compatible serial interface for standard (100 kHz), fast (400 kHz), or high speed (3.4 MHz) mode.

The MCP4725 is an ideal DAC device where design simplicity and small footprint is desired, and for applications requiring the DAC device settings to be saved during power-off time.

The device is available in a small 6-pin SOT-23 package.

Package Type

SOT-23-6



PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	XX	X	X	/XX
Device	Address Options	Tape and Reel	Temperature Range	Package

Device: MCP4725: Single Channel 12-Bit DAC w/EEPROM Memory

Address Options:

XX		A2	A1	A0
A0 *	=	0	0	External
A1	=	0	1	External
A2	=	1	0	External
A3	=	1	1	External

* Default option. Contact Microchip factory for other address options

Tape and Reel: T = Tape and Reel

Temperature Range: E = -40°C to +125°C

Package: CH = Plastic Small Outline Transistor (SOT-23-6), 6-lead

Examples:

a) MCP4725A0T-E/CH: Tape and Reel, Extended Temp., 6LD SOT-23 pkg. Address Option = A0

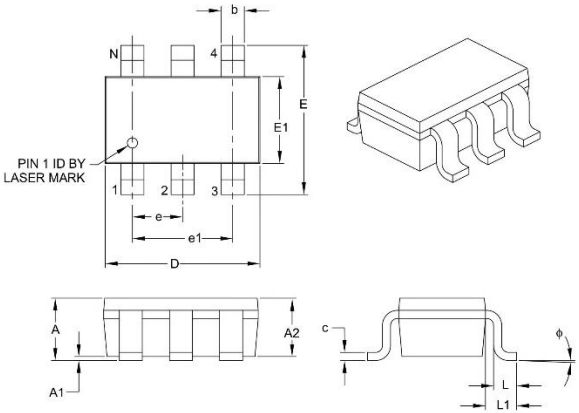
b) MCP4725A1T-E/CH: Tape and Reel, Extended Temp., 6LD SOT-23 pkg. Address Option = A1

c) MCP4725A2T-E/CH: Tape and Reel, Extended Temp., 6LD SOT-23 pkg. Address Option = A2

d) MCP4725A3T-E/CH: Tape and Reel, Extended Temp., 6LD SOT-23 pkg. Address Option = A3

6-Lead Plastic Small Outline Transistor (CH) [SOT-23]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		6	
Pitch	e		0.95 BSC	
Outside Lead Pitch	e1		1.90 BSC	
Overall Height	A	0.90	–	1.45
Molded Package Thickness	A2	0.89	–	1.30
Standoff	A1	0.00	–	0.15
Overall Width	E	2.20	–	3.20
Molded Package Width	E1	1.30	–	1.80
Overall Length	D	2.70	–	3.10
Foot Length	L	0.10	–	0.60
Footprint	L1	0.35	–	0.80
Foot Angle	ϕ	0°	–	30°
Lead Thickness	c	0.08	–	0.26
Lead Width	b	0.20	–	0.51

Notes:

1. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.127 mm per side.
2. Dimensioning and tolerancing per ASME Y14.5M.

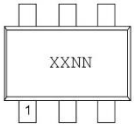
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-028B

10.0 PACKAGING INFORMATION

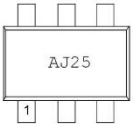
10.1 Package Marking Information

6-Lead SOT-23



Part Number	Address Option	Code
MCP4725A0T-E/CH	A0 (00)	AJNN
MCP4725A1T-E/CH	A1 (01)	APNN
MCP4725A2T-E/CH	A2 (10)	AQNN
MCP4725A3T-E/CH	A3 (11)	ARNN

Example



Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

9.0 DEVELOPMENT SUPPORT

9.1 Evaluation & Demonstration Boards

The MCP4725 SOT-23-6 Evaluation Board is available from Microchip Technology Inc. This board works with Microchip's PICkit™ Serial Analyzer. The user can program the DAC input codes and EEPROM data, or read the programmed data using the easy to use PICkit Serial Analyzer with the Graphic User Interface software. Refer to www.microchip.com for further information on this product's capabilities and availability.

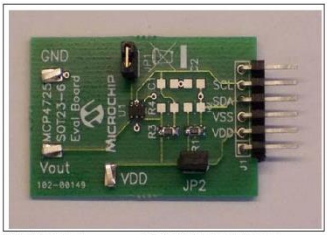


FIGURE 9-1: *MCP4725 SOT-23-6 Evaluation Board.*

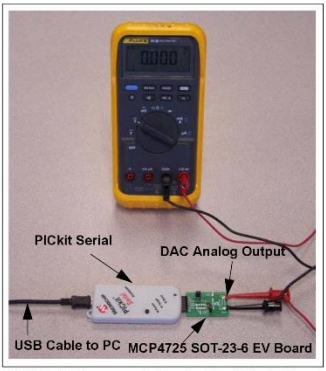


FIGURE 9-2: *Setup for the MCP4725 SOT-23-6 Evaluation Board with PICkit™ Serial Analyzer.*

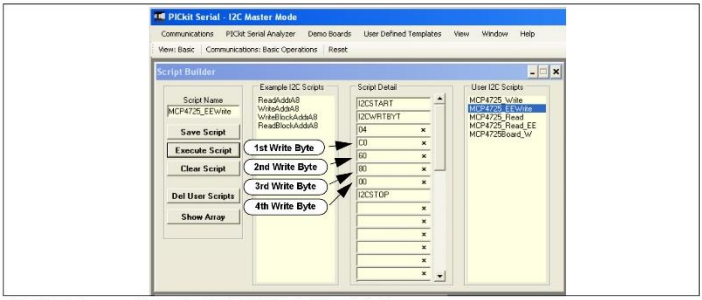


FIGURE 9-3: *Example of PICkit™ Serial User Interface.*

8.5.4.1 Design a Bipolar DAC using Example 8-3

Some applications desires an output step magnitude of 1 mV with an output range of ±2.05V. The following steps explain the design solution:

1. Calculate the range: +2.05V – (-2.05V) = 4.1V.
2. Calculate the resolution needed:
   4.1V/1 mV = 4100

   Since 2^{12} = 4096 for 12-bit resolution.
3. The amplifier gain (R_2/R_1), multiplied by V_{DD}, must be equal to the desired minimum output to achieve bipolar operation. Since any gain can be realized by choosing resistor values (R_1+R_2), the V_{DD} value must be selected first. If a V_{DD} of 4.1V is used, solve for the amplifier's gain by setting the DAC to 0, knowing that the output needs to be -2.05V. The equation can be simplified to:

$$\frac{-R_2}{R_1} = \frac{-2.05}{V_{DD}} = \frac{-2.05}{4.1} \rightarrow \frac{R_2}{R_1} = \frac{1}{2}$$

If R_1 = 20 kΩ and R_2 = 10 kΩ, the gain will be 0.5.

4. Next, solve for R_3 and R_4 by setting the DAC to 4096, knowing that the output needs to be +2.05V.

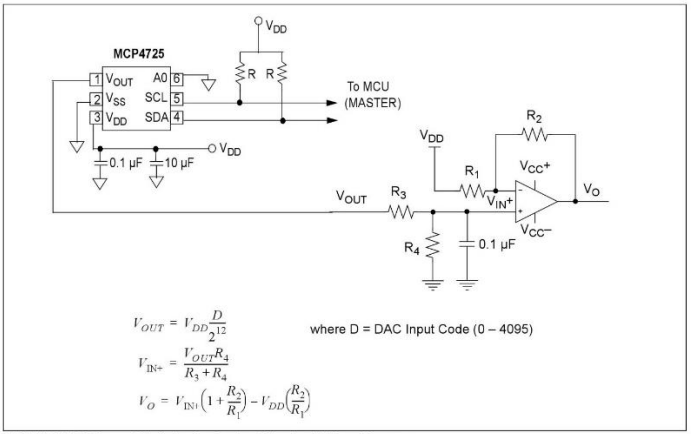
$$\frac{R_4}{(R_3+R_4)} = \frac{2.05V + (0.5 \cdot V_{DD})}{1.5 \cdot V_{DD}} = \frac{2}{3}$$

If R_4 = 20 kΩ, then R_3 = 10 kΩ

8.5.4 BIPOLAR OPERATION

Bipolar operation is achievable using the MCP4725 by using an external operational amplifier (op amp). This allows a general purpose DAC, with its cost and availability advantages, to meet almost any desired output voltage range, power and noise performance.

Example 8-3 illustrates a simple bipolar voltage source configuration. R_1 and R_2 allow the gain to be selected, while R_3 and R_4 shift the DAC's output to a selected offset. Note that R4 can be tied to V_{DD} (= V_{REF}) instead of V_{SS}, if a higher offset is desired. Note that a pull-up to V_{DD} could be used, instead of R_4, if a higher offset is desired.



$$V_{OUT} = V_{DD}\frac{D}{2^{12}}$$

where D = DAC Input Code (0 – 4095)

$$V_{IN+} = \frac{V_{OUT}R_4}{R_3 + R_4}$$

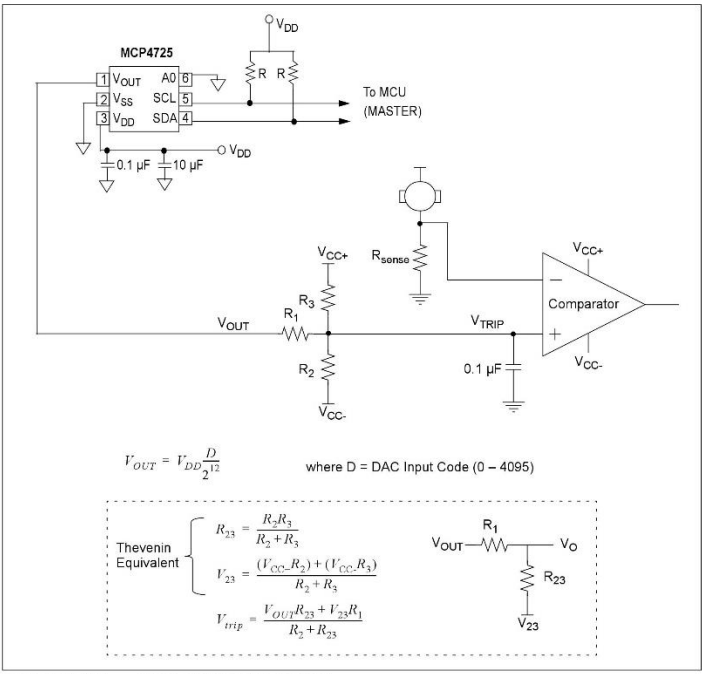
$$V_O = V_{IN+}\left(1 + \frac{R_2}{R_1}\right) - V_{DD}\left(\frac{R_2}{R_1}\right)$$

EXAMPLE 8-3: *Digitally-Controlled Bipolar Voltage Source.*

8.5.3 BUILDING A "WINDOW" DAC

Some sensor applications require very high resolution around the set point or threshold voltage.

Example 8-2 shows an example of creating a "window" around the threshold using a voltage divider network with a pull-up and pull-down resistor. In the circuit, the output voltage range is scaled down, but its step resolution is increased greatly.



$$V_{OUT} = V_{DD}\frac{D}{2^{12}}$$

$$R_{23} = \frac{R_2 R_3}{R_2 + R_3}$$

$$V_{23} = \frac{(V_{CC} \cdot R_2) + (V_{CC} \cdot R_3)}{R_2 + R_3}$$

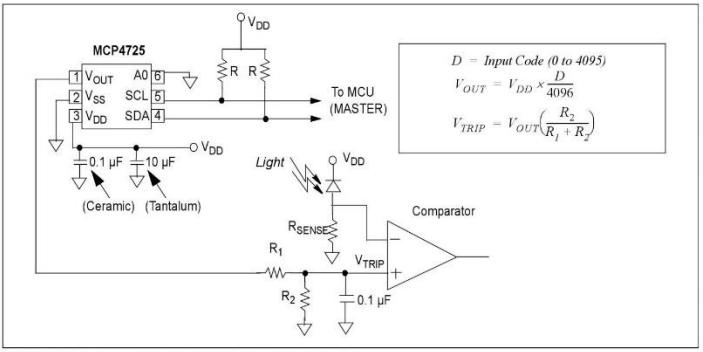
$$V_{trip} = \frac{V_{OUT} R_{23} + V_{23} R_1}{R_1 + R_{23}}$$

EXAMPLE 8-2: *Single-Supply "Window" DAC.*

8.5.2 DECREASING THE OUTPUT STEP SIZE

Calibrating the threshold of a diode, transistor or resistor may require a very small step size in the DAC output voltage. These applications may require about 200 µV of step resolution within 0.8V of range.

One method of achieving this small step resolution is using a voltage divider at the DAC output. An example is shown in Example 8-1. The step size of the DAC output is scaled down by the factor of the ratio of the voltage divider. Note that the bypass capacitor on the output of the voltage divider plays a critical function in attenuating the output noise of the DAC and the induced noise from the environment.



EXAMPLE 8-1: *Set Point Or Threshold Calibration.*

8.2 Using Non-Volatile EEPROM Memory

The user can store the DAC input code (12 bits) and power-down configuration bits (2 bits) in the internal non-volatile EEPROM memory using the I^2C write command. The user can also read the EEPROM data using the I^2C read command. When the device is first powered after power is shut down, the device uploads the EEPROM contents to the DAC register automatically and provides the DAC output immediately. This feature is very useful in applications where the DAC device is used to provide set point or calibration data for other devices in the application system. The DAC will not lose the important system operational parameters due to the system power failure incidents. See **Section 5.6 "Non-Volatile EEPROM Memory"** for more details of the non-volatile EEPROM memory.

8.3 Power Supply Considerations

The power supply to the device is used for both V_{DD} and DAC reference voltage. Any noise induced on the V_{DD} line can affect on the DAC performance. Typical application will require a bypass capacitor in order to filter out high frequency noise on the V_{DD} line. The noise can be induced onto the power supply's traces or as a result of changes on the DAC output. The bypass capacitor helps to minimize the effect of these noise sources on signal integrity. Figure 8-1 shows an example of using two bypass capacitors (a 10 µF tantalum capacitor and a 0.1 µF ceramic capacitor) in parallel on the V_{DD} line. These capacitors should be placed as close to the V_{DD} pin as possible (within 4 mm).

The power source should be as clean as possible. If the application circuit has separate digital and analog power supplies, the V_{DD} and V_{SS} pins of the MCP4725 should reside on the analog plane.

8.4 Layout Considerations

Inductively-coupled AC transients and digital switching noise from other devices can affect on DAC performance and DAC output signal integrity. Careful board layout will minimize these effects. Bench testing has shown that a multi-layer board utilizing a low-inductance ground plane, isolated inputs, isolated outputs and proper decoupling are critical to achieving the performance that the MCP4725 is capable of providing. Particularly harsh environments may require shielding of critical signals. Separate digital and analog ground planes are recommended. In this case, the V_{SS} pin and the ground pins of the V_{DD} capacitors of the MCP4725 should be terminated to the analog ground plane.

8.5 Application Examples

The MCP4725 is a rail-to-rail output DAC designed to operate with a V_{DD} range of 2.7V to 5.5V. Its output amplifier is robust enough to drive common, small-signal loads directly, thus eliminating the cost and size of an external buffer for most applications.

8.5.1 DC SET POINT OR CALIBRATION

A common application for the MCP4725 is a digitally-controlled set point or a calibration of variable parameters such as sensor offset or bias point. Example 8-1 shows an example of the set point setting. Since the MCP4725 is a 12-bit DAC and uses the V_{DD} supply as a reference source, it provides a V_{DD}/4096 of resolution per step.

8.0 TYPICAL APPLICATIONS

The MCP4725 device is one of Microchip's latest DAC device family with non-volatile EEPROM memory. The device is a general purpose resistive string DAC intended to be used in applications where a precision, and low power DAC with moderate bandwidth is required.

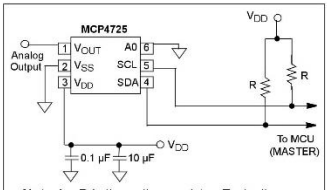
Since the device includes non-volatile EEPROM memory, the user can use this device for applications that require the output to return to the previous set-up value on subsequent power-ups.

Applications generally suited for the MCP4725 device family include:

- Set Point or Offset Trimming
- Sensor Calibration
- Portable Instrumentation (Battery Powered)
- Motor Speed Control

8.1 Connecting to I²C BUS using Pull-Up Resistors

The SCL and SDA pins of the MCP4725 are open-drain configurations. These pins require a pull-up resistor as shown in Figure 8-1. The value of these pull-up resistors depends on the operating speed (standard, fast, and high speed) and loading capacitance of the I²C bus line. Higher value of pull-up resistor consumes less power, but increases the signal transition time (higher RC time constant) on the bus. Therefore, it can limit the bus operating speed. The lower resistor value, on the other hand, consumes higher power, but allows higher operating speed. If the bus line has higher capacitance due to long bus line or high number of devices connected to the bus, a smaller pull-up resistor is needed to compensate the long RC time constant. The pull-up resistor is typically chosen between 1 kΩ and 10 kΩ ranges for standard and fast modes, and less than 1 kΩ for high speed mode.



Note 1: R is the pull-up resistor. Typically 1 ~ 10 kΩ

2: A0 can be tied to V_{SS}, V_{DD} or driven by MCU

FIGURE 8-1: *I²C Bus Interface Connection with A0 pin tied to V_{SS}.*

Two devices with the same A2 and A1 address bits can be connected to the same I²C bus by utilizing the A0 address pin (Example: A0 pin of device A is tied to V_{DD}, and the other device's pin is tied to Vss.)

8.1.1 DEVICE CONNECTION TEST

The user can test the presence of the MCP4725 on the I²C bus line without performing the data conversion. This test can be achieved by checking an acknowledge response from the MCP4725 after sending a read or write command. Here is an example using Figure 8-2:

(a) Set the R/$\overline{W}$ bit "HIGH" in the address byte.

(b) If the MCP4725 is connected to the I²C bus line, it will then acknowledge by pulling SDA bus LOW during the ACK clock and then release the bus back to the I²C Master.

(c) A STOP or repeated START bit can then be issued from the Master and I²C communication can continue.

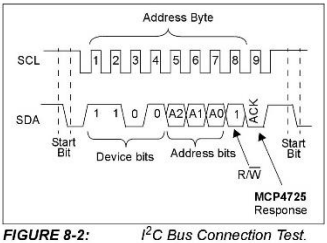


FIGURE 8-2: *I²C Bus Connection Test.*

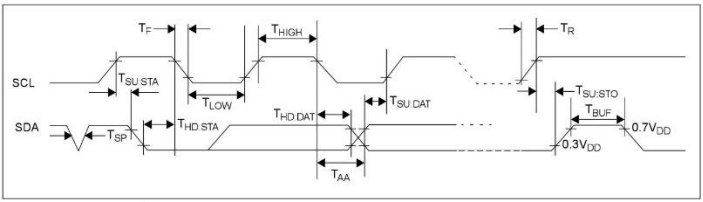


FIGURE 7-4: *I^2C Bus Timing Data.*

TABLE 7-1: I²C SERIAL TIMING SPECIFICATIONS (CONTINUED)

Electrical Specifications: Unless otherwise specified, all limits are specified for T_A = -40 to +85°C, V_{DD} = +2.7V to +5.0V, V_{SS} = 0V.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
High Speed Mode (Note 5)						
Clock frequency	f_{SCL}	0	—	3.4 1.7	MHz MHz	C_b = 100 pF C_b = 400 pF
Clock high time	T_{HIGH}	60 120	—	—	ns ns	C_b = 100 pF C_b = 400 pF
Clock low time	T_{LOW}	160 320	—	—	ns	C_b = 100 pF C_b = 400 pF
SCL rise time **(Note 1)**	T_R	—	—	40 80	ns	From V_{IL} to V_{IH}, C_b = 100 pF C_b = 400 pF
SCL fall time **(Note 1)**	T_F	—	—	40 80	ns	From V_{IH} to V_{IL}, C_b = 100 pF C_b = 400 pF
SDA rise time **(Note 1)**	$T_{R:DAT}$	—	—	80 160	ns	From V_{IL} to V_{IH}, C_b = 100 pF C_b = 400 pF
SDA fall time **(Note 1)**	$T_{F:DATA}$	—	—	80 160	ns	From V_{IH} to V_{IL}, C_b = 100 pF C_b = 400 pF
START condition hold time	$T_{HD:STA}$	160	—	—	ns	After this period, the first clock pulse is generated
Repeated START condition setup time	$T_{SU:STA}$	160	—	—	ns	Only relevant for repeated Start condition
Data hold time **(Note 4)**	$T_{HD:DAT}$	0 0	—	70 150	ns	C_b = 100 pF C_b = 400 pF
Data input setup time	$T_{SU:DAT}$	10	—	—	ns	
STOP condition setup time	$T_{SU:STO}$	160	—	—	ns	
STOP condition hold time	$T_{HD:STO}$	160	—	—	ns	
Output valid from clock **(Notes 2 and 3)**	T_{AA}	—	—	150 310	ns	C_b = 100 pF C_b = 400 pF
Bus free time	T_{BUF}	160	—	—	ns	Time between START and STOP conditions.

Note 1: This parameter is ensured by characterization and not 100% tested.

2: This specification is not a part of the I2C specification. This specification is equivalent to the Data Hold Time ($T_{HD:DAT}$) plus SDA Fall (or rise) time: $T_{AA} = T_{HD:DAT} + T_F$ (OR T_R).

3: If this parameter is too short, it can create an unintended Start or Stop condition to other devices on the bus line. If this parameter is too long, Clock Low time (T_{LOW}) can be affected.

4: For Data Input: This parameter must be longer than t_{SP}. If this parameter is too long, the Data Input Setup ($T_{SU:DAT}$) or Clock Low time (T_{LOW}) can be affected.
For Data Output: This parameter is characterized, and tested indirectly by testing T_{AA} parameter.

5: All timing parameters in high-speed modes are tested at V_{DD} = 5V.

TABLE 7-1: I^2C SERIAL TIMING SPECIFICATIONS

Electrical Specifications: Unless otherwise specified, all limits are specified for T_A = -40 to +85°C, V_{DD} = +2.7V to +5.0V, V_{SS} = 0V.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Standard Mode						
Clock frequency	f_{SCL}	0	—	100	kHz	
Clock high time	T_{HIGH}	4000	—	—	ns	
Clock low time	T_{LOW}	4700	—	—	ns	
SDA and SCL rise time **(Note 1)**	T_R	—	—	1000	ns	From V_{IL} to V_{IH}
SDA and SCL fall time **(Note 1)**	T_F	—	—	300	ns	From V_{IH} to V_{IL}
START condition hold time	$T_{HD:STA}$	4000	—	—	ns	After this period, the first clock pulse is generated.
Repeated START condition setup time	$T_{SU:STA}$	4700	—	—	ns	Only relevant for repeated Start condition
Data hold time **(Note 3)**	$T_{HD:DAT}$	0	—	3450	ns	
Data input setup time	$T_{SU:DAT}$	250	—	—	ns	
STOP condition setup time	$T_{SU:STO}$	4000	—	—	ns	
STOP condition hold time	$T_{HD:STD}$	4000	—	—	ns	
Output valid from clock **(Notes 2 and 3)**	T_{AA}	0	—	3750	ns	
Bus free time	T_{BUF}	4700	—	—	ns	Time between START and STOP conditions.
Fast Mode						
Clock frequency	T_{SCL}	0	—	400	kHz	
Clock high time	T_{HIGH}	600	—	—	ns	
Clock low time	T_{LOW}	1300	—	—	ns	
SDA and SCL rise time **(Note 1)**	T_R	20 + 0.1Cb	—	300	ns	From V_{IL} to V_{IH}
SDA and SCL fall time **(Note 1)**	T_F	20 + 0.1Cb	—	300	ns	From V_{IH} to V_{IL}
START condition hold time	$T_{HD:STA}$	600	—	—	ns	After this period, the first clock pulse is generated
Repeated START condition setup time	$T_{SU:STA}$	600	—	—	ns	Only relevant for repeated Start condition
Data hold time **(Note 4)**	$T_{HD:DAT}$	0	—	900	ns	
Data input setup time	$T_{SU:DAT}$	100	—	—	ns	
STOP condition setup time	$T_{SU:STO}$	600	—	—	ns	
STOP condition hold time	$T_{HD:STD}$	600	—	—	ns	
Output valid from clock **(Notes 2 and 3)**	T_{AA}	0	—	1200	ns	
Bus free time	T_{BUF}	1300	—	—	ns	Time between START and STOP conditions.

Note 1: This parameter is ensured by characterization and not 100% tested.

2: This specification is not a part of the I2C specification. This specification is equivalent to the Data Hold Time ($T_{HD:DAT}$) plus SDA Fall (or rise) time: $T_{AA} = T_{HD:DAT} + T_F$ (OR T_R).

3: If this parameter is too short, it can create an unintended Start or Stop condition to other devices on the bus line. If this parameter is too long, Clock Low time (T_{LOW}) can be affected.

4: For Data Input: This parameter must be longer than t_{SP}. If this parameter is too long, the Data Input Setup ($T_{SU:DAT}$) or Clock Low time (T_{LOW}) can be affected.
For Data Output: This parameter is characterized, and tested indirectly by testing T_{AA} parameter.

5: All timing parameters in high-speed modes are tested at V_{DD} = 5V.

7.5.5 ACKNOWLEDGE

Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

The device that acknowledges, has to pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. During reads, a master must send an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave.

In this case, the slave (MCP4725) will leave the data line HIGH to enable the master to generate the STOP condition.

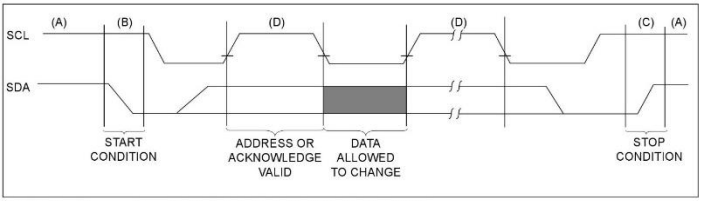


FIGURE 7-3: *Data Transfer Sequence On The Serial Bus.*

7.0 I²C SERIAL INTERFACE COMMUNICATION

7.1 OVERVIEW

The MCP4725 device uses a two-wire I²C serial interface that can operate on a standard, fast or high speed mode. A device that sends data onto the bus is defined as transmitter, and a device receiving data as receiver. The bus has to be controlled by a master device which generates the serial clock (SCL), controls the bus access and generates the START and STOP conditions. The MCP4725 device works as slave. Both master and slave can operate as transmitter or receiver, but the master device determines which mode is activated. An example of hardware connection diagram is shown in Figure 8-1. Communication is initiated by the master (microcontroller) which sends the START bit, followed by the slave address byte. The first byte transmitted is always the slave address byte, which contains the device code, the address bits, and the R/$\overline{W}$ bit. The device code for the MCP4725 device is 1100.

When the device receives a read command (R/W = 1), it transmits the contents of the DAC input register and EEPROM. A non-acknowledge (NAK) or repeated start bit can be transmitted at any time. See Figure 6-3 for the read operation example. If writing to the device (R/W = 0), the device will expect write command type bits in the following byte. See Figure 6-1 and Figure 6-2 for the write operation examples.

The MCP4725 supports all three I²C operating modes:

- Standard Mode: bit rates up to 100 kbit/s
- Fast Mode: bit rates up to 400 kbit/s
- High Speed Mode (HS mode): bit rates up to 3.4 Mbit/s

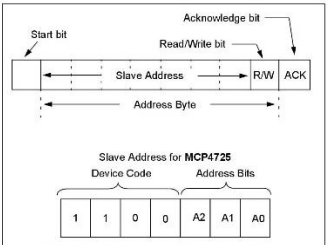
Refer to the Phillips I²C document for more details of the I²C specifications.

7.2 Device Addressing

The address byte is the first byte received following the START condition from the master device. The first part of the address byte consists of a 4-bit device code which is set to 1100 for the MCP4725. The device code is followed by three address bits (A2, A1, A0) which are programmed as follows:

- The choice of A2 and A1 bits are provided by the customer as part of the ordering process. These bits are then programmed (hard-wired) during manufacturing
- The A2 and A1 are programmed to '00' (default), if not requested by customer
- A0 bit is determined by the logic state of A0 pin. The A0 pin can be tied to V_{DD} or V_{SS}, or can be actively driven by digital logic levels. The advantage of using the A0 pin is that the users can control the A0 bit on their application PCB circuit and also two identical MCP4725 devices can be used on the same bus line.

When the device receives an address byte, it compares the logic state of the A0 pin with the A0 address bit received before responding with the acknowledge bit. The logic state of the A0 pin needs to be set prior to the interface communication.



Note: A2 and A1: Programmed (hard-wired) at the factory. Please Contact Microchip Technology Inc. for A2 and A1 programming options.
A0: Use the logic level state of A0 pin.

FIGURE 7-1: *Device Addressing*

6.2 READ COMMAND

If the R/$\overline{W}$ bit is set to a logic "high", then the device outputs on SDA pin, the DAC register and EEPROM data. Figure 6-3 shows an example of reading the register and EEPROM data. The 2nd byte in Figure 6-3 indicates the current condition of the device operation. The RDY/$\overline{BSY}$ bit indicates EEPROM writing status. The RDY/$\overline{BSY}$ bit stays low during EEPROM writng and high when the writing is completed..

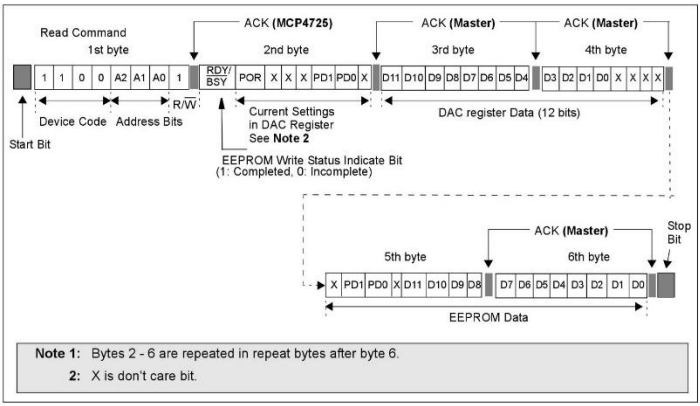


FIGURE 6-3: *Read Command and Output Data Format.*

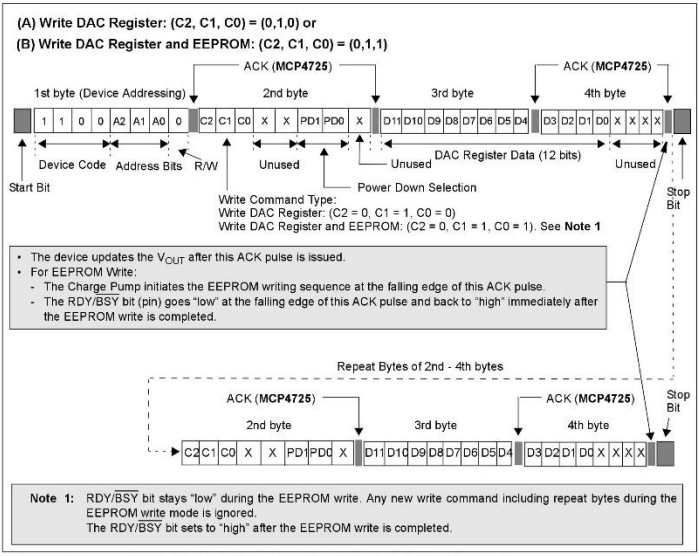


FIGURE 6-2: *Write Commands for DAC Input Register and EEPROM.*

TABLE 6-2: WRITE COMMAND TYPE

C2	C1	C0	Command Name	Function
0	0	X	Fast Mode	This command is used to change the DAC register. EEPROM is not affected
0	0	X	"	"
0	1	0	Write DAC Register	Load configuration bits and data code to the DAC Register
0	1	1	Write DAC Register and EEPROM	(a) Load configuration bits and data code to the DAC Register and (b) also write the EEPROM
1	0	0	Reserved	Reserved for future use
1	0	1	Reserved	Reserved for future use
1	1	0	Reserved	Reserved for future use
1	1	1	Reserved	Reserved for future use

Note 1: X = Dont' Care. Fast Mode does not use C0 bit.
2: The MCP4725 ignores the "Reserved" commands.

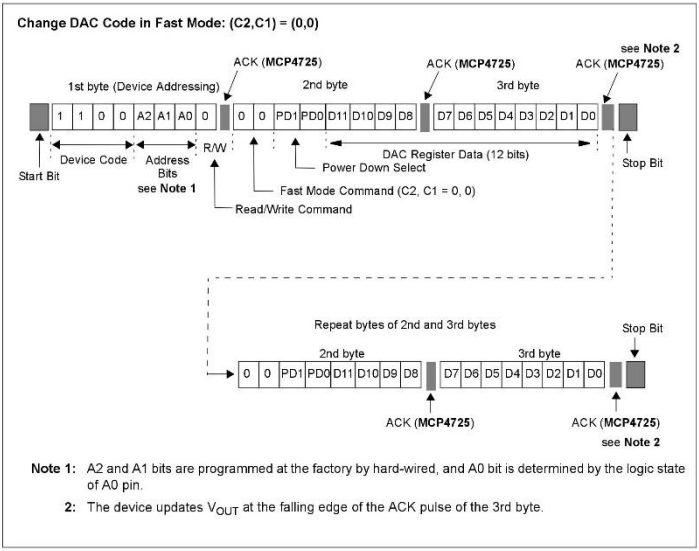


FIGURE 6-1: *Write Command for Fast Mode.*

6.0 THEORY OF OPERATION

When the device is connected to the I^2C bus line, the device is working as a slave device. The Master (MCU) can write/read the DAC input register or EEPROM using the I^2C interface command. The MCP4725 device address contains four fixed bits (1100 = device code) and three address bits (A2, A1, A0). The A2 and A1 bits are hard-wired during manufacturing, and A0 bit is determined by the logic state of A0 pin. The A0 pin can be connected to V_{DD} or V_{SS}, or actively driven by digital logic levels.

The following sections describe the communication protocol to send or read the data code and write/read the EEPROM using the I^2C interface. See **Section 7.0 "I^2C Serial Interface Communication"**.

6.1 Write Commands

The write commands are used to load the configuration bits and DAC input code to the DAC register, or to write to the EEPROM of the device. The write command types are defined by using three write command type bits (C2, C1, C0). Table 6-2 shows the write command types and their functions. There are three command types for the MCP4725. The four "reserved" commands in Table 6-2 are for future use. The MCP4725 ignores the "reserved" commands. Write command protocol examples are shown in Figure 6-1 and Figure 6-2.

The input data code is coded as shown in Table 6-1. The MSB of the data is always transmitted first and the format is unipolar binary.

TABLE 6-1: INPUT DATA CODING

Input Code	Nominal Output Voltage (V)
1111111111111 (FFFh)	V_{DD} - 1 LSB
1111111111110 (FFEh)	V_{DD} - 2 LSB
0000000000010 (002h)	2 LSB
0000000000001 (001h)	1 LSB
0000000000000 (000h)	0

6.1.1 WRITE COMMAND FOR FAST MODE (C2 = 0, C1 = 0, C0 = X, X = DON'T CARE)

The fast write command is used to update the DAC register. The data in the EEPROM of the device is not affected by this command. This command updates Power-Down mode selection bits (PD1 and PD0) and 12 bits of the DAC input code in the DAC register. Figure 6-1 shows an example of the fast write command for the MCP4725 device.

6.1.2 WRITE COMMAND FOR DAC INPUT REGISTER (C2 = 0, C1 = 1, C0 = 0)

In MCP4725, this command performs the same function as the Fast Mode command in **Section 6.1.1 "Write Command for Fast mode (C2 = 0, C1 = 0, C0 = X, X = Don't Care)"**. Figure 6-2 shows the write command protocol for the MCP4725.

As shown in Figure 6-2, the D11 - D0 bits in the third and fourth bytes are DAC input data. The last 4 bits (X, X, X, X) in the fourth byte are don't care bits.

The device executes the Master's write command after receiving the last byte (4th byte). The Master can send a STOP bit to terminate the current sequence, or send a Repeated START bit followed by an address byte. If the device receives three data bytes continuously after the 4th byte, it updates from the 2nd to the 4th data bytes with the last three input data bytes.

The contents of the register are updated at the end of the 4th byte. The device ignores any partially received data bytes if the I^2C communication with the Master ends before completing the 4th byte.

6.1.3 WRITE COMMAND FOR DAC INPUT REGISTER AND EEPROM (C2 = 0, C1 = 1, C0 = 1)

When the device receives this command, it (a) loads the configuration and data bits to the DAC register, and (b) also writes the EEPROM. When the device is writing the EEPROM, the RDY/$\overline{BSY}$ bit goes low and stays low until the EEPROM write operation is completed. The state of the RDY/$\overline{BSY}$ bit can be monitored by a read command. Figure 6-2 shows the details of the this write command protocol and Figure 6-3 shows the details of the read command.

5.6 Non-Volatile EEPROM Memory

The MCP4725 device has a 14-bit wide EEPROM memory to store configuration bit (2 bits) and DAC input data (12 bits). These bits are readable and re-writable with I^2C interface commands. The device has an on-chip charge pump circuit to write the EEPROM memory bits without using an external program voltage.

The EEPROM writing operation is initiated when the device receives an EEPROM write command (C2 = 0, C1 = 1, C0 = 1). The configuration and writing data bits are transferred to the EEPROM memory block. A status bit, RDY/$\overline{BSY}$, stays low during the EEPROM writing and goes high as the write operation is completed. While the RDY/$\overline{BSY}$ bit is low (during the EEPROM writing), any new write command is ignored (for EEPROM or DAC register). Table 5-3 shows the EEPROM bits and factory default settings. Table 5-4 shows the DAC input register bits of the MCP4725.

TABLE 5-3: EEPROM MEMORY AND FACTORY DEFAULT SETTINGS (TOTAL NUMBER OF BITS: 14 BITS)

Bit Name	PD1	PD0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit Function	Power-Down Select (2 bits)		DAC Input Data (12 bits)											
Factory Default Value	0	0 (1)	1 (2)	0	0	0	0	0	0	0	0	0	0	0

Note 1: See Table 5-2 for details.

2: Bit D11 = '1' (while all other bits are "0") enables the device to output 0.5 * V_{DD} (= middle scale output).

TABLE 5-4: DAC REGISTER

Bit Name	C2	C1	C0	RDY/$\overline{BSY}$	POR	PD1	PD0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit Function	Command Type			(1)		Power-Down Select		Data (12 bits)											

Note 1: Write EEPROM status indication bit (0:EEPROM write is not completed. 1:EEPROM write is complete.)

5.5 Normal and Power-Down Modes

The device has two modes of operation: Normal mode and power-down mode. The mode is selected by programming the power-down bits (PD1 and PD0) in the Configuration register. The user can also program the two power-down bits in non-volatile EEPROM memory.

When the normal mode is selected, the device operates a normal digital-to-analog conversion. If the power-down mode is selected, the device enters a power saving condition by shutting down most of the internal circuits. During the power-down mode, all internal circuits except the I^2C interface are disabled and there is no data conversion event, and no V_{OUT} is available. The device also switches the output stage from the output of the amplifier to a known resistive load. The value of the resistive load is determined by the state of the power-down bits (PD1 and PD0). Table 5-2 shows the outcome of the power-down bit and the resistive load.

During the power-down mode, the device draws about 60 nA (typical). Although most of internal circuits are shutdown, the serial interface remains active in order to receive the I^2C command.

The device exits the power-down mode immediately when (a) it receives a new write command for normal mode or (b) it receives an I^2C General Call Wake-Up Command.

When the DAC operation mode is changed from power-down to normal mode, the output settling time takes less than 10 µs, but greater than the standard Active mode settling time (6 µs, typical).

TABLE 5-2: POWER-DOWN BITS

PD1	PD0	Function
0	0	Normal Mode
0	1	1 kΩ resistor to ground (1)
1	0	100 kΩ resistor to ground (1)
1	1	500 kΩ resistor to ground (1)

Note 1: In the power-down mode: V_{OUT} is off and most of internal circuits are disabled.

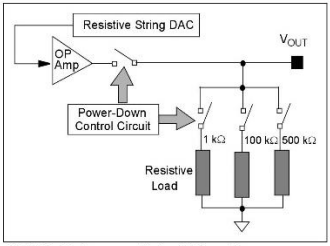


FIGURE 5-1: *Output Stage for Power-Down Mode.*

5.0 GENERAL DESCRIPTION

The MCP4725 is a single channel buffered voltage output 12-bit DAC with non-volatile memory (EEPROM). The user can store configuration register bits (2 bits) and DAC input data (12 bits) in non-volatile EEPROM (14 bits) memory.

When the device is powered on first, it loads the DAC code from the EEPROM and outputs the analog output accordingly with the programmed settings. The user can reprogram the EEPROM or DAC register any time.

The device uses a resistor string architecture. DAC's output is buffered with a low power precision amplifier. This output amplifier provides low offset voltage and low noise, as well as rail-to-rail output. The amplifier can also provide high source currents (V_{OUT} pin to V_{SS}).

The DAC can be configured to normal or power saving power-down mode by setting the configuration register bits.

The device uses a two-wire I^2C compatible serial interface and operates from a single power supply ranging from 2.7V to 5.5V.

5.1 Output Voltage

The input coding to the MCP4725 device is unsigned binary. The output voltage range is from 0V to V_{DD}. The output voltage is given in Equation 5-1:

EQUATION 5-1:

$$V_{OUT} = \frac{(V_{REF} \times D_n)}{4096}$$

Where:

V_{REF} = V_{DD}
D_n = Input code

5.1.1 OUTPUT AMPLIFIER

The DAC output is buffered with a low-power, precision CMOS amplifier. This amplifier provides low offset voltage and low noise. The output stage enables the device to operate with output voltages close to the power supply rails. Refer to **Section 1.0 "Electrical Characteristics"** for range and load conditions.

The output amplifier can drive the resistive and high capacitive loads without oscillation. The amplifier can provide maximum load current as high as 25 mA which is enough for most of a programmable voltage reference applications.

5.1.2 DRIVING RESISTIVE AND CAPACITIVE LOADS

The MCP4725 output stage is capable of driving loads up to 1000 pF in parallel with 5 kΩ load resistance. Figure 2-15 shows the V_{OUT} vs. Resistive Load. V_{OUT} drops slowly as the load resistance decreases after about 3.5 kΩ.

5.2 LSB SIZE

One LSB is defined as the ideal voltage difference between two successive codes. (see Equation 4-1). Table 5-1 shows an example of the LSB size over full-scale range (V_{DD}).

TABLE 5-1: LSB SIZES FOR MCP4725 (EXAMPLE)

Full-Scale Range (V_{DD})	LSB Size	Condition
3.0V	0.73 mV	3 / 4096
5.0V	1.22 mV	5 / 4096

5.3 Voltage Reference

The MCP4725 device uses the V_{DD} as its voltage reference. Any variation or noises on the V_{DD} line can affect directly on the DAC output. The V_{DD} needs to be as clean as possible for accurate DAC performance.

5.4 Reset Conditions

In the Reset conditions, the device uploads the EEPROM data into the DAC register. The device can be reset by two independent events: (a) by POR or (b) by I^2C General Call Reset Command.

The factory default settings for the EEPROM prior to shipment are shown in Table 4-3 (set for a middle scale output). The user can rewrite or read the DAC register or EEPROM anytime after the Power-On-Reset event.

5.4.1 POWER-ON-RESET

The device's internal Power-On-Reset (POR) circuit ensures that the device powers up in a defined state.

If the power supply voltage is less than the POR threshold (V_{POR} = 2V, typical), all circuits are disabled and there will be no DAC output. When the V_{DD} increases above the V_{POR}, the device takes a reset state. During the reset period, the device uploads all configuration and DAC input codes from EEPROM. The DAC output will be the same as for the value last stored in the EEPROM. This enables the device returns to the same state that it was at the last write to the EEPROM before it was powered off.

4.9 Offset Error Drift

Offset error drift is the variation in offset error due to a change in ambient temperature. The offset error drift is typically expressed in ppm/°C.

4.10 Settling Time

The Settling time is the time delay required for the DAC output to settle to its new output value from the start of code transition, within specified accuracy. In the MCP4725, the settling time is a measure of the time delay until the DAC output reaches its final value (within 0.5 LSB) when the DAC code changes from 400h to C00h.

4.11 Major-Code Transition Glitch

Major-code transition glitch is the impulse energy injected into the DAC analog output when the code in the DAC register changes state. It is normally specified as the area of the glitch in nV-Sec. and is measured when the digital code is changed by 1 LSB at the major carry transition (Example: 011...111 to 100... 000, or 100... 000 to 011 ... 111).

4.12 Digital Feedthrough

Digital feedthrough is the glitch that appears at the analog output caused by coupling from the digital input pins of the device. It is specified in nV-Sec. and is measured with a full scale change on the digital input pins (Example: 000... 000 to 111... 111, or 111... 111 to 000... 000). The digital feedthrough is measured when the DAC is not being written to the register.

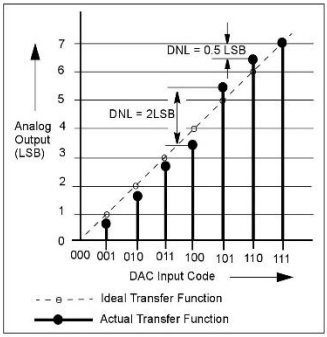


FIGURE 4-2: *DNL Accuracy.*

4.5 Offset Error

Offset error (Figure 4-3) is the deviation from zero voltage output when the digital input code is zero. This error affects all codes by the same amount. In the MCP4725, the offset error is not trimmed at the factory. However, it can be calibrated by software in application circuits.

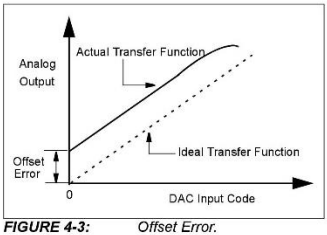


FIGURE 4-3: *Offset Error.*

4.6 Gain Error

Gain error (see Figure 4-4) is the difference between the actual full-scale output voltage from the ideal output voltage on the transfer curve. The gain error is calculated after nullifying the offset error, or full scale error minus the offset error.

The gain error indicates how well the slope of the actual transfer function matches the slope of the ideal transfer function. The gain error is usually expressed as percent of full-scale range (% of FSR) or in LSB.

In the MCP4725, the gain error is not calibrated at the factory and most of the gain error is contributed by the output op amp saturation near the code range beyond 4000. For the applications which need the gain error specification less than 1% maximum, the user may consider using the DAC code range between 100 and 4000 instead of using full code range (code 0 to 4095). The DAC output of the code range between 100 and 4000 is much linear than full-scale range (0 to 4095). The gain error can be calibrated by software in applications.

4.7 Full-Scale Error (FSE)

Full-scale error (Figure 4-4) is the sum of offset error plus gain error. It is the difference between the ideal and measured DAC output voltage with all bits set to one (DAC input code = FFFh).

EQUATION 4-4:

$$FSE = \frac{(V_{OUT} - V_{Ideal})}{LSB}$$

Where:

V_{Ideal} = $(V_{REF})\,(1 - 2^{-n}) - V_{OFFSET}$

V_{REF} = The reference voltage. $V_{REF} = V_{DD}$ in the MCP4725

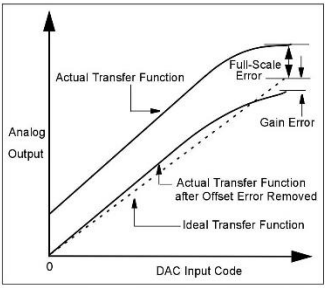


FIGURE 4-4: *Gain Error and Full-Scale Error.*

4.8 Gain Error Drift

Gain error drift is the variation in gain error due to a change in ambient temperature. The gain error drift is typically expressed in ppm/°C.

3.0 PIN DESCRIPTIONS

The descriptions of the pins are listed in Table 3-1.

TABLE 3-1: PIN FUNCTION TABLE

Pin No. SOT-23	Name	Function
1	V_{OUT}	Analog Output Voltage
2	V_{SS}	Ground Reference
3	V_{DD}	Supply Voltage
4	SDA	I^2C Serial Data
5	SCL	I^2C Serial Clock Input
6	A0	Device Address Selection pin. This pin can be tied to V_{SS} or V_{DD}, or can be actively driven by the digital logic levels. The logic state of this pin determines what the A0 bit of the I^2C address bits should be.

3.1 Analog Output Voltage (V_{OUT})

V_{OUT} is an analog output voltage from the DAC device. DAC output amplifier drives this pin with a range of V_{SS} to V_{DD}.

3.2 Supply Voltage (V_{DD}, V_{SS})

V_{DD} is the power supply pin for the device. The voltage at the V_{DD} pin is used as the supply input as well as the DAC reference input. The power supply at the V_{DD} pin should be clean as possible for a good DAC performance.

This pin requires an appropriate bypass capacitor of about 0.1 µF (ceramic) to ground. An additional 10 µF capacitor (tantalum) in parallel is also recommended to further attenuate high frequency noise present in application boards. The supply voltage (V_{DD}) must be maintained in the 2.7V to 5.5V range for specified operation.

V_{SS} is the ground pin and the current return path of the device. The user must connect the V_{SS} pin to a ground plane through a low impedance connection. If an analog ground path is available in the application PCB (printed circuit board), it is highly recommended that the V_{SS} pin be tied to the analog ground path or isolated within an analog ground plane of the circuit board.

3.3 Serial Data Pin (SDA)

SDA is the serial data pin of the I^2C interface. The SDA pin is used to write or read the DAC register and EEPROM data. The SDA pin is an open-drain N-channel driver. Therefore, it needs a pull-up resistor from the V_{DD} line to the SDA pin. Except for start and stop conditions, the data on the SDA pin must be stable during the high period of the clock. The high or low state of the SDA pin can only change when the clock signal on the SCL pin is low. Refer to **Section 7.0 "I^2C Serial Interface Communication"** for more details of I^2C Serial Interface communication.

3.4 Serial Clock Pin (SCL)

SCL is the serial clock pin of the I^2C interface. The MCP4725 acts only as a slave and the SCL pin accepts only external serial clocks. The input data from the Master device is shifted into the SDA pin on the rising edges of the SCL clock and output from the MCP4725 occurs at the falling edges of the SCL clock. The SCL pin is an open-drain N-channel driver. Therefore, it needs a pull-up resistor from the V_{DD} line to the SCL pin. Refer to **Section 7.0 "I^2C Serial Interface Communication"** for more details of I^2C Serial Interface communication.

3.5 Device Address Selection Pin (A0)

This pin is used to select the A0 address bit by the user. The user can tie this pin to V_{SS} (logic '0'), or V_{DD} (logic '1'), or can be actively driven by the digital logic levels, such as the I^2C Master Output. See **Section 7.2 "Device Addressing"** for more details of the address bits.

Note: Unless otherwise indicated, T_A = +25°C, V_{DD} = +5.0V, V_{SS} = 0V, R_L = 5 kΩ to V_{SS}, C_L = 100 pF.

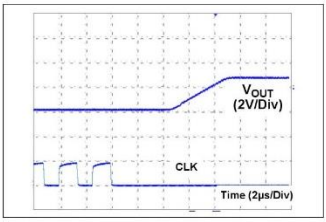


FIGURE 2-24: *Exiting Power Down Mode.*

Note: Unless otherwise indicated, T_A = +25°C, V_{DD} = +5.0V, V_{SS} = 0V, R_L = 5 kΩ to V_{SS}, C_L = 100 pF.

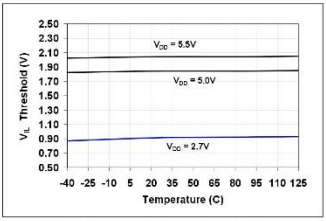


FIGURE 2-18: *V_{IN} Low Threshold vs. Temperature and V_{DD}.*

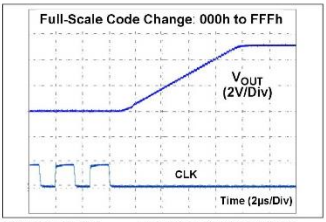


FIGURE 2-19: *Full-Scale Settling Time.*

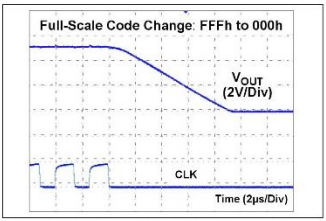


FIGURE 2-20: *Full-Scale Settling Time.*

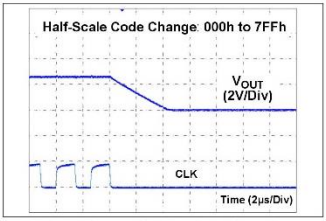


FIGURE 2-21: *Half-Scale Settling Time.*

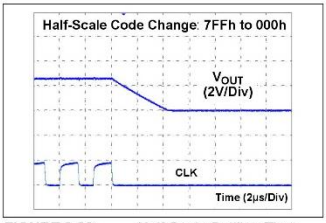


FIGURE 2-22: *Half-Scale Settling Time.*

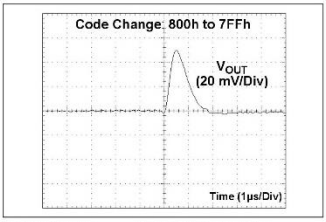


FIGURE 2-23: *Code Change Glitch.*

Note: Unless otherwise indicated, T_A = +25°C, V_{DD} = +5.0V, V_{SS} = 0V, R_L = 5 kΩ to V_{SS}, C_L = 100 pF.

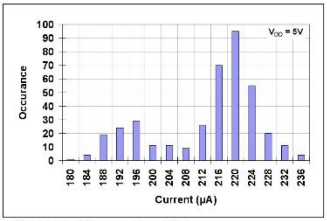


FIGURE 2-12: *I_{DD} Histogram .*

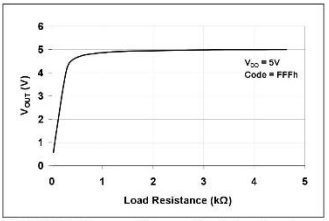


FIGURE 2-15: *V_{OUT} vs. Resistive Load.*

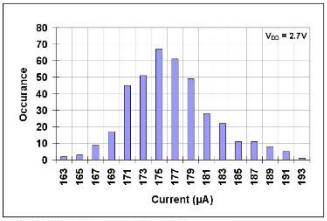


FIGURE 2-13: *I_{DD} Histogram.*

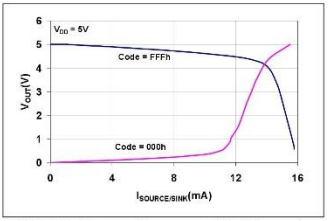


FIGURE 2-16: *Source and Sink Current Capability.*

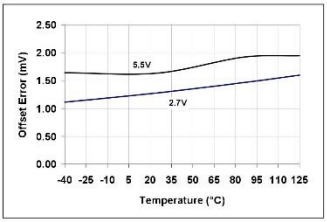


FIGURE 2-14: *Offset Error vs. Temperature and V_{DD}.*

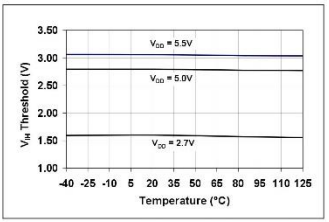


FIGURE 2-17: *V_{IN} High Threshold vs. Temperature and V_{DD}.*

Note: Unless otherwise indicated, T_A = +25°C, V_{DD} = +5.0V, V_{SS} = 0V, R_L = 5 kΩ to V_{SS}, C_L = 100 pF.

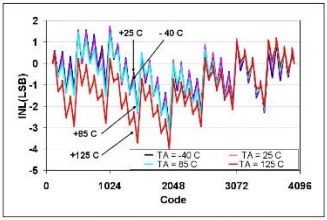


FIGURE 2-7: *INL vs. Code and Temperature (V_{DD} = 2.7V).*

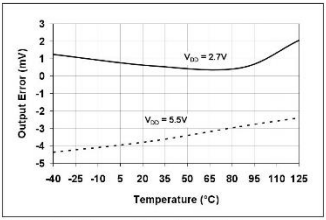


FIGURE 2-10: *Output Error vs. Temperature (Code = 4000d).*

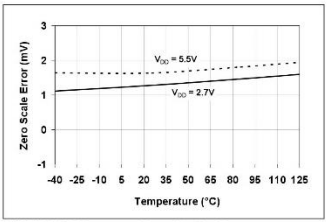


FIGURE 2-8: *Zero Scale Error vs. Temperature (Code = 000d).*

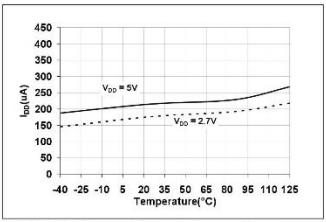


FIGURE 2-11: *I_{DD} vs. Temperature.*

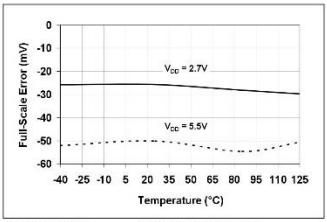


FIGURE 2-9: *Full-Scale Error vs. Temperature (Code = 4095d).*

2.0 TYPICAL PERFORMANCE CURVES

Note:	The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

Note: Unless otherwise indicated, T_A = +25°C, V_{DD} = +5.0V, V_{SS} = 0V, R_L = 5 kΩ to V_{SS}, C_L = 100 pF.

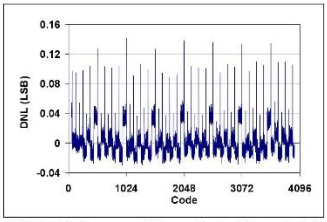


FIGURE 2-1: *DNL vs. Code (V_{DD} = 5.5V).*

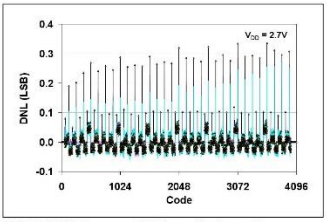


FIGURE 2-4: *DNL vs. Code and Temperature (T_A = -40°C to +125°C).*

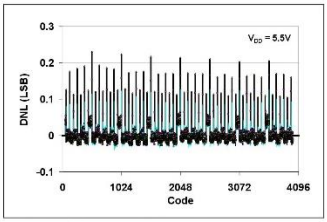


FIGURE 2-2: *DNL vs. Code and Temperature (TA = -40°C to +125°C).*

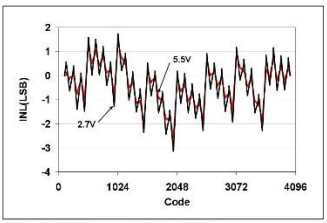


FIGURE 2-5: *INL vs. Code.*

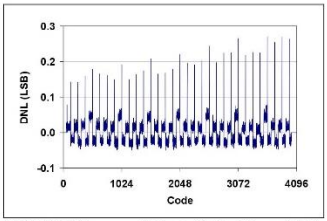


FIGURE 2-3: *DNL vs. Code (V_{DD} = 2.7V).*

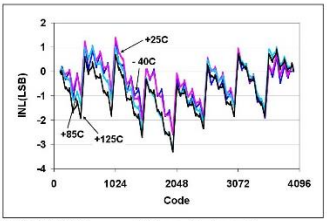


FIGURE 2-6: *INL vs. Code and Temperature (V_{DD} = 5.5V).*

TEMPERATURE CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = +2.7V to +5.5V, V_{SS} = GND.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Temperature Ranges						
Specified Temperature Range	T_A	-40	—	+125	°C	
Operating Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, 6L-SOT-23	θ_{JA}	—	190	—	°C/W	

ELECTRICAL CHARACTERISTICS (CONTINUED)

Electrical Specifications: Unless otherwise indicated, all parameters apply at V_{DD} = + 2.7V to 5.5V, V_{SS} = 0V, R_L = 5 kΩ from V_{OUT} to V_{SS}, C_L = 100 pF, T_A = -40°C to +125°C. Typical values are at +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Power Up Time	T_{PU}	—	2.5	—	µs	V_{DD} = 5V
		—	5	—	µs	V_{DD} = 3V Coming out of Power-down mode, started from falling edge of ACK pulse in I^2C command.
DC Output Impedance	R_{OUT}	—	1	—	Ω	Normal mode (V_{OUT} to V_{SS})
		—	1	—	kΩ	Power-Down Mode 1 (V_{OUT} to V_{SS})
		—	100	—	kΩ	Power-Down Mode 2 (V_{OUT} to V_{SS})
		—	500	—	kΩ	Power-Down Mode 3 (V_{OUT} to V_{SS})
Dynamic Performance						
Major Code Transition Glitch		—	45	—	nV-s	1 LSB change around major carry (800h to 7FFh) (**Note 2**)
Digital Feedthrough		—	<10	—	nV-s	**Note 2**
Digital Interface						
Output Low Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 3 mA
Input High Voltage (SDA and SCL Pins)	V_{IH}	$0.7V_{DD}$	—	—	V	
Input Low Voltage (SDA and SCL Pins)	V_{IL}	—	—	$0.3V_{DD}$	V	
Input High Voltage (A0 Pin)	$V_{A0\text{-}IH}$	$0.8V_{DD}$	—	—		**Note 4**
Input Low Voltage (A0 Pin)	$V_{A0\text{-}IL}$	—	—	$0.2V_{DD}$		**Note 4**
Input Leakage	I_{LI}	—	—	±1	µA	SCL = SDA = A0 = V_{SS} or SCL = SDA = A0 = V_{DD}
Pin Capacitance	C_{PIN}	—	—	3	pF	**Note 2**
EEPROM						
EEPROM Write Time	TWRITE	—	25	50	ms	EEPROM Write time for 14 bits
Data Retention		—	200	—	Years	At +25°C, **(Note 2)**
Endurance		1	—	—	Million Cycles	At +25°C, **(Note 2)**

Note 1: Test Code Range: 100 to 4000.

2: This parameter is ensure by design and not 100% tested.

3: Within 1/2 LSB of the final value when code changes from 1/4 to 3/4 (400h to C00h) of full-scale.

4: Logic state of external address pin (A0 pin).

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings†

V_{DD}.....6.5V
All inputs and outputs w.r.t V_{SS} .....–0.3V to V_{DD}+0.3V
Current at Input Pins .....±2 mA
Current at Supply Pins .....±50 mA
Current at Output Pins .....±25 mA
Storage Temperature .....-65°C to +150°C
Ambient Temp. with Power Applied .....-55°C to +125°C
ESD protection on all pins .....≥ 6 kV HBM, ≥ 400V MM
Maximum Junction Temperature (T_J) .....+150°C

† Notice: Stresses above those listed under "Maximum ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability

ELECTRICAL CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, all parameters apply at V_{DD} = + 2.7V to 5.5V, V_{SS} = 0V, R_L = 5 kΩ from V_{OUT} to V_{SS}, C_L = 100 pF, T_A = -40°C to +125°C. Typical values are at +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Operating Voltage	V_{DD}	2.7		5.5	V	
Supply Current	I_D	—	210	400	µA	Digital input grounded, output unloaded, code = 000h
Power-Down Current	I_{DDP}	—	0.06	2.0	µA	V_{DD} = 5.5V
Power-On-Reset Threshold	V_{POR}	—	2	—	V	
DC Accuracy						
Resolution	n	12	—	—	Bits	Code Range = 000h to FFFh
INL Error	INL	—	±2	±14.5	LSB	**Note 1**
DNL	DNL	-0.75	±0.2	±0.75	LSB	**Note 1**
Offset Error	V_{OS}		0.02	0.75	% of FSR	Code = 000h
Offset Error Drift	ΔV_{OS}/°C	—	±1	—	ppm/°C	-45°C to +25°C
		—	±2	—	ppm/°C	+25°C to +85°C
Gain Error	G_E	-2	-0.1	2	% of FSR	Code FFFh, not including offset error
Gain Error Drift	ΔG_E/°C	—	-3	—	ppm/°C	
Output Amplifier						
Phase Margin	p_M	—	66	—	Degree(°)	C_L = 400 pF, R_L = ∞
Capacitive Load Stability	C_L	—	—	1000	pF	R_L = 5 kΩ, **Note 2**
Slew Rate	SR	—	0.55	—	V/µs	
Short Circuit Current	I_{SC}	—	15	24	mA	V_{DD} = 5V, V_{OUT} = Grounded
Output Voltage Settling Time	T_S	—	6	—	µs	**Note 3**

Note 1: Test Code Range: 100 to 4000.
2: This parameter is ensure by design and not 100% tested.
3: Within 1/2 LSB of the final value when code changes from 1/4 to 3/4 (400h to C00h) of full-scale.
4: Logic state of external address pin (A0 pin).

Biodata

Nama : Fajar Alif Chalifatullah
Jenis Kelamin : Pria
Tempat Lahir : Surabaya
Tanggal Lahir : 1 Februari 1998
Agama : Islam
Alamat : Dupak Rukun III/27 Surabaya
HP : 089678759356
E-mail : fajaralifc535 @gmail.com

Riwayat Pendidikan

2015 – 2018 : Departemen Teknik Elektro Otomasi - Fakultas Vokasi – Institut Teknologi Sepuluh Nopember (ITS) Surabaya.
2012 - 2015 : SMAN 21 Surabaya.
2010 - 2012 : SMPN 42 Surabaya.
2004 - 2010 : SDN Tembok Dukuh Kawasan Surabaya.

“Halaman Ini Sengaja Dikosongkan”