**Position Control of Manipulator using PMAC with DC Servo Motors**

**ORGANIZATION PROFILE**

The **“Defence Research and Development Laboratory” (DRDL**) in Hyderabad, formerly directed by A.P.J. Abdul Kalam, is the main research center for the Integrated Missile Development Program. It is located in the Defence Research Complex at Kanchanbagh, on the periphery of Hyderabad Old City.

****“Defence Research & Development Laboratory” (D.R.D.L**) is one of the laboratories of DRDO, which** was formed in 1958 from the amalgamation of the then already Functioning Technical Development Establishment (TDEs) of the Indian Army and the Directorate of Technical Development & Production (DTDP) with the Defence Science Organization (DSO). DRDO was then a small organization with 10 establishments or laboratories. Over the years, it has grown multi-directionally in terms of the variety of subject disciplines, number of laboratories, achievements and stature.

Today, DRDO is a network of 51 laboratories, which are deeply engaged in developing defence technologies covering various disciplines, like aeronautics, armaments, electronics, combat vehicles, engineering systems, instrumentation, missiles, advanced computing and simulation, special materials, naval systems, life sciences, training, information systems and agriculture. Presently, the Organization is backed by over 5000 scientists and about 25,000 other scientific, technical and supporting personnel.

Several major projects for the development of missiles, armaments, light combat aircrafts, radars, electronic warfare systems etc are on hand and significant achievements have already been made in several such technologies.

**ABSTRACT**

The Delta-Tau Systems, Inc. Programmable Multi-Axis Controller(PMAC) is a family of high performance servo motion controllers capable of commanding upto eight axes of motion simultaneously with a high level of sophistication. Through the power of a Digital Signal Processor (DSP), PMAC offers a price-performance ratio for multi-axis control that was not previously available. Motorola’s DSP 56001 is the CPU for PMAC, and it handles all the calculations for all eight axes.

As a general purpose controller , PMAC can serve in a wide variety of applications, from those requiring sub-microinch precision to those needing hundreds of kilowatts or horsepower. Its diverse uses include robotics, machine tools, paper and lumber processing, assembly lines, food processing, printing, packaging, material handling, camera control, automatic welding, silicon water processing, laser cutting and many others.

The project work aims at controlling the position of DC servomotor using DC servo drive. **PC based position controlling is done for single or multi-axis. Desired Position Command is given to the PMAC through PEWIN32PRO software which gives its output to Servo Drive to drive the servomotor at specific speed, direction and position. Out of the four hardware versions, the non-turbo type PMAC-PCI Lite is used.**

****CHAPTER 1: OVERVIEW****

**1.1 Introduction**

This project is aimed to demonstrate the feasibility of integrating the DC servomotor to the clamping mechanism and controlling its position through Programmable Multi Axis Controller called PMAC with Dc servo drive.

**1.2 Aim of the project**

* Integration of DC servomotor to clamp mechanism with DC servo drive.
* Interfacing of Programmable Multi Axis Controller to the DC servo drives in closed loop.
* Parameter optimization and Tuning of the drive.

**1.3 Methodology**

As the system employ the PC based stand alone controller, the precise positioning of the table become possible with the soft CNC, motion control program and PLC for I/O control of the system.

The controller involved in this application is Programmable Multi Axis Controller (PMAC). It is a high performance servo motion controller capable of commanding up to 8 axes of motion simultaneously with a high level of sophistication. Motorola’s DSP 56001 is the CPU for PMAC, and it handles all the calculations for 8 axes.

The servo drive used for this application is transistor controlled PWM DC servo drive. By employing servo drive we can get effective position control. With this speed can also be controlled from 0.1 rpm to full rated speed.

**1.4 Flexibility and Configuration for a task**

There are four hardware versions of PMAC: the PMAC-PC, the PMAC-Lite, the PMAC-VME, and the PMAC-STD. These cards differ from each other in their form factor, the nature of the bus interface, and in the availability of certain I/O ports. All versions of the card have identical on-board firmware, so PMAC programs written for one version will run on any other version. As a general-purpose controller, PMAC can serve in a wide variety of applications.

PMAC is configured for a particular application by the choice of the hardware set(through options and accessories), configuration of parameters, and the writing of motion and PLC programs. Each PMAC possess firmware capable of controlling eight axes. The eight axes can be all associated together for completely coordinated motion; each axis can be put in its own coordinate system for eight completely independent operations. The PMAC CPU communicates with the axes through specially designed custom gate array ICs, refered to as “DSPGATES”. Each of these ICs can handle four analog output channels, four encoders as input, and four analog-derived inputs from accessories board.Upto 16 PMAC may be ganged together with complete synchronization, for a total of 128axes.

**1.6 PMAC is a Computer**

It is important to note that PMAC is a full computer in its own right, capable of standalone operation with its own stored programs. Furthermore, it is a real-time, multitasking computer that can prioritize tasks and have the higher priority tasks pre-empt those of lower priority (most personal computers are not capable of this). Even when used with a host computer, the communications should be thought of as those from one computer to another, not as computer to peripheral. In these applications , the ability to run multiple tasks simultaneously,properly prioritized can take a tremendous burden off the host computer (and its programmer!), both in terms of time, and of task-switching complexity.

**CHAPTER 2 : BACKGROUND STUDY**

**2.1 Computer Numerical control (CNC)**

The abbreviation **CNC** stands for **C**omputer(ized) **N**umerical(ly) **C**ontrol(led*)*, and refers specifically to the computer control of [machine tools](http://en.wikipedia.org/wiki/Machine_tool) for the purpose of (repeatedly) [manufacturing](http://en.wikipedia.org/wiki/Manufacturing) complex parts in [metal](http://en.wikipedia.org/wiki/Metal) as well as other [materials](http://en.wikipedia.org/wiki/Material), using a [program](http://en.wikipedia.org/wiki/Computer_program). CNC was developed in the late [1940s](http://en.wikipedia.org/wiki/1940s) and early [1950s](http://en.wikipedia.org/wiki/1950s). CNC machines were relatively briefly preceded by the less advanced NC, or Numerically Controlled machines.

**2.1.1 Numerical Control (NC)**

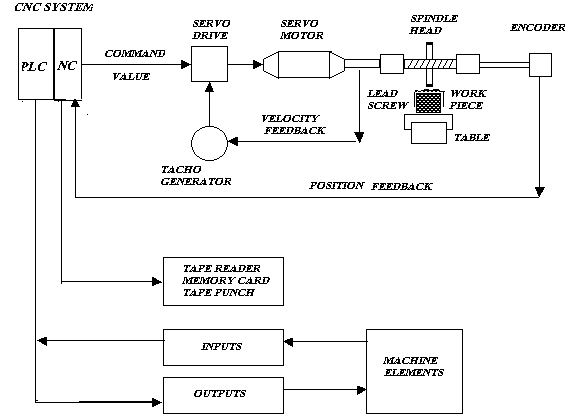
Numerical control or numerically controlled (NC) machine tools are machines that are automatically operated by commands that are received by their processing units. NC machines were first developed soon after World War II and made it possible for large quantities of the desired components to be very precisely and efficiently produced in a reliable repetitive manner.

Numerical Control (NC) was the precursor of today's Computer Numerical Control (CNC), which controls the automation of machine tools and the inherent tool processes for which they are designed. The CNC machine tool is the servo actuator of the [CAD/CAM](http://en.wikipedia.org/wiki/CAD/CAM) (Computer Assisted Design/Computer Assisted Manufacturing) technology both literally and figuratively. CNC inherits from NC the essential character of by-the-numbers interpolation of transition points in the work envelope of a mult-axis motion platform, based on the separation of programming from operations. Once stored in the CNC memory and selected, the program is executed by pressing the appropriate key on the machine operator panel.

The axes movements of all CNC machines are controlled in closed loop. Closed loop provides position control with feedback. Feedback minimizes the difference between commanded position and the actual position and also the commanded velocity to the actual velocity

A closed loop motion –control system consists of a controller (the CNC System), an amplifier to drive motor, a motor an transducer that monitors the motion. The position loop terminates onto the CNC System and the CNC System generates the Analog command voltage to the amplifier. The velocity loop terminates in the driver amplifier.

**FIG A : Flow chart showing CNC system**



* 1. **CNC System Features**
* Axis, Spindle and I/O Control Feature
* Programming features
* Key board display features
* Communication features
* Compensation features
* Safety and Diagnostics features

**2.3 Sub Systems of CNC system**

**2.3.1 Axes Controller**

Depending on the distance to be moved for the axis and the feed rate, the micro commands are calculated. This error signal (i.e., the command voltage) is the difference of the micro command & position feedback.The primary function of the feed drive is to cause the motion of the controlled machine tool member (Table, slide etc.,) conform as closely as possible to the motion commands issued by the NC system.

**2.3.2 Spindle Controller**

The position feed back of the spindle is for operation and for the control of the axis with respect to the spindle rpm in specific machine functions. The spindle relation is controlled by the axis controller subsystem in some systems.

**2.3.3 I/O Controller or PLC**

This controls the ON/OFF functions of machine tool. It sets the outputs based on input conditions & corresponding logistics.

**2.4 SERVO SYSTEM**

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio controlled cars, puppets, and of course, robots. Servos are extremely useful in robotics. The motors are small, as you can see by the picture above, have built in control circuitry, and are extremely powerful for their size.

[](http://www.seattlerobotics.org/guide/images/servo3c.jpg)

**A servo disassembled.**

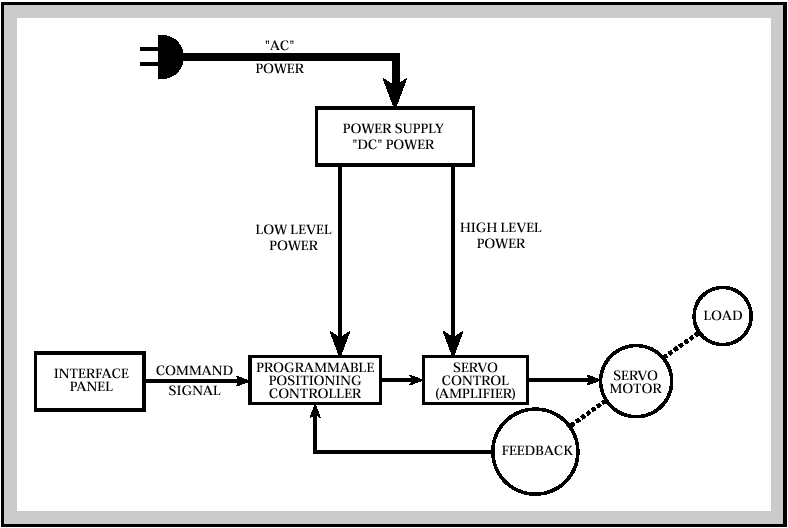
[](http://www.seattlerobotics.org/guide/images/servo1.jpg)

**A Futaba S-148 Servo**

The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. In the picture above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, its somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

A command signal, which is issued from, the user's interface panel, comes into the servo's "positioning controller" which is a device that stores information about various jobs or tasks. It has been programmed to activate the motor/load, i.e. change speed/position. The signal then passes into the servo control or "amplifier" section. The servo control takes this low power level signal and increases, or amplifies the power up to appropriate levels to actually result in movement of the servo motor/load. These low power level signals must be amplified. Higher voltage levels are needed to rotate the servomotor at appropriate higher speeds and higher current levels are required to provide torque to move heavier loads. As power is applied onto the servomotor, the load begins to move . . . speed and position changes. As the load moves, so does some other "device" move. This other "device" is either a Tachometer, resolver or encoder (providing a signal which is "sent back" to the controller). This "feedback" signal is informing the positioning controller whether the motor is doing the proper job.

# Fig 2.2 Concept of a Servo System



**2.5 Servo Motor**

Basic construction and operation principles of the servomotor are the same as general conventional induction motors. But they have been redesigned to meet high precision, high speed, high frequency positioning and speed control of mechanical facilities. Servomotors are classified into DC servomotors, AC servomotors, and stepping motors.

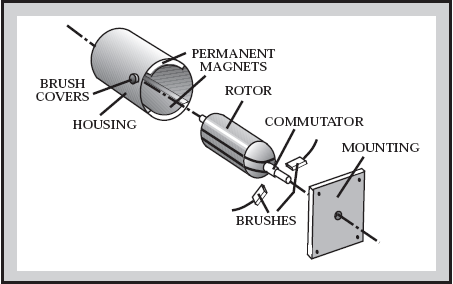
**2.5.1 DC Servo Motor Features**

* Capacity (watt): Less than 500 W
* Advantages
  + - * Smaller outside dimensions and large torque
      * Good operating efficiency.
      * Good controllability
      * Cheap.

**2.5.2 PMDC Motor**

The predominant motor configuration utilized in demanding incrementing (start-stop) applications is the permanent magnet DC (PMDC) motor. This type with appropriate feedback is quite an effective device in closed loop servo system applications. Since the stator field is generated by permanent magnets, no power is used for field generation. The magnets provide constant field flux at all speeds. Therefore, linear speed torque curves result. This motor type provides relatively high starting, or acceleration torque, is linear and predictable, and has a smaller frame and lighter weight compared to other motor types and provides rapid positioning.

**Fig 2.3 Typical DC motor construction**



**2.6 Feedback Devices**

Feedback devices are used to measure the position of the slide and close the control loop. The accuracy of positioning of the slide is largely dependent on the resolution of the feedback device in CNC machine. There are two types of feedback systems such as

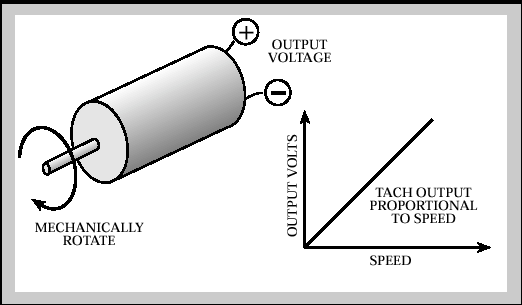
**1. Velocity Feedback system.**

**2. Position Feedback System.**

Servos use feedback signals for stabilization, speed and position information. This information may come from a variety of devices such as the analog tachometer, the digital tachometer (optical encoder) or resolver.

**2.6.1 Analog Tachometer**

The shaft is turned by some mechanical means and a voltage is developed at the terminals. The faster the shaft is turned, the larger the magnitude of voltage developed i.e. the amplitude of the tacho signal is directly proportional to speed. The output voltage shows a polarity (+ or -), which is dependent on direction of rotation. They play an important role in drives, because of their ability to provide directional and rotational information. They can be used to provide speed information to a meter (for visual speed readings) or provide velocity feedback (for stabilization purposes).



**2.6.2 Digital Tachometer**

A digital tachometer, often termed an optical encoder or simply encoder, is a mechanical-to-electrical conversion device. The encoder's shaft is rotated and an output signal results which is proportional to distance (i.e. angle) the shaft is rotated through. The output signal may be square waves, or sinusoidal waves, or provide an absolute position. Thus encoders are classified into two basic types: absolute and incremental.

**2.6.3 Encoders**

Machines that move need a means of measuring movement. Since the machine tools, inspection machines, material handling equipment and the like have themselves evolved from basic rudimentary manual machines to highly sophisticated automated pieces, so have the internal measuring mechanisms. The most common type of measurement component today is the encoder. Encoders detect the flashes of light that come shining through a slotted disk attached to the rotating shaft.

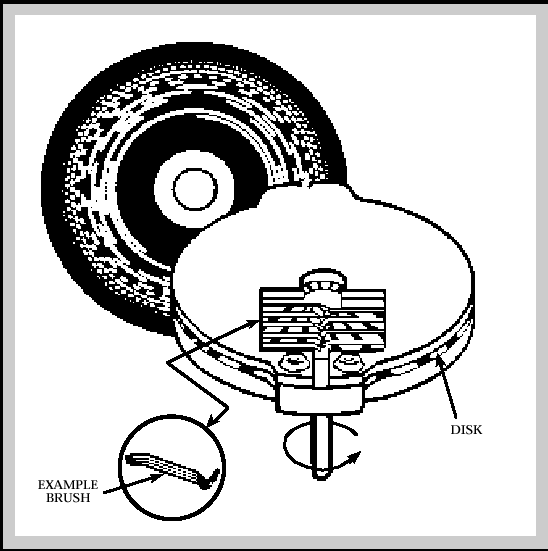
Encoders can be generally categorized into optical (photoelectric), magnetic encoders, and mechanical contact types. Photoelectric encoders in particular—due to their high accuracy, high reliability and relatively low cost, play a significant role in machine tool technology.



**2.6.4 Absolute Encoder**

The absolute encoder provides a specific address for each shaft position throughout 360 degrees. This type of encoder employs either contact (brush) or non-contact schemes of sensing position.

The contact scheme incorporates a brush assembly to make direct electrical contact with the electrically conductive paths of the coded disk to read address information. The non-contact scheme utilizes photoelectric detection to sense position of the coded disk. The number of tracks on the coded disk may be increased until the desired resolution or accuracy is achieved.



**Fig 2.5 Absolute Encoder**

And since position information is directly on the coded disk built-in "memory system" and a power failure will not cause this information to be lost. Therefore, it will not be required to return to a "home" or "start" position upon re-energizing power.

**2.6.5 Incremental Encoder**

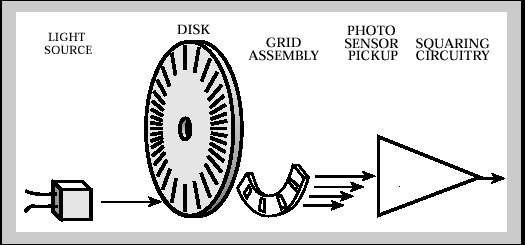
The incremental encoder provides either pulses or a sinusoidal output signal as it is rotated throughout 360 degrees. Thus distance data is obtained by counting this information. The disk is manufactured with opaque lines. A light source passes a beam through the transparent segments onto a photo sensor, which outputs a sinusoidal waveform. Electronic processing can be used to transform this signal into a square pulse train. In utilizing this device, the following parameters are important:

1) **Line count:** This is the number of pulses per revolution. The number of lines is determined by the positional accuracy required in the application.

2) **Output signal:** The output from the photo sensor can be either a sine or square wave signal.

3) **Number of channels:** Either one or two channel out-puts can be provided.

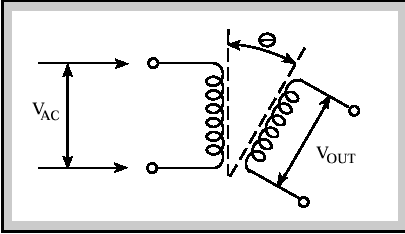
**Fig 2.6 Incremental Encoder**



**2.6.6 Resolvers**

Resolvers look similar to small motors – that is, one end has terminal wires, and the other end has a mounting flange and a shaft extension. Internally, a "signal" winding rotor revolves inside a fixed stator. This represents a type of trans-former: When one winding is excited with a signal, through transformer action the second winding is excited. As the first winding is moved (the rotor), the out-put of the second winding changes (the stator). This change is directly proportion-al to the angle, which the rotor has been moved through. As a starting point, the simplest resolver unit contains a single winding on the rotor and two windings on the stator (located 90 degrees apart). A reference signal is applied onto the primary (the rotor), then via transformer action this is coupled to the secondary. The secondary output signal would be a sine wave proportional to angle.

### Fig 2.7 Resolver



### 

**2.7 Uses of CNC machines:**

The introduction of CNC machines radically changed the [manufacturing](http://en.wikipedia.org/wiki/Manufacturing) industry. Curves are as easy to cut as straight lines, complex 3-D structures are relatively easy to produce, and the number of [machining](http://en.wikipedia.org/wiki/Machining) steps that required human action have been dramatically reduced.

With the increased automation of manufacturing processes with CNC [machining](http://en.wikipedia.org/wiki/Machining), considerable improvements in consistency and quality have been achieved. CNC automation reduced the frequency of errors and provided CNC operators with time to perform additional tasks.

CNC automation also allows for more flexibility in the way parts are held in the manufacturing process and the time required to change the machine to produce different components.

In a production environment, a series of CNC machines may be combined into one station, commonly called a "cell", to progressively machine a part requiring several operations.

**CHAPTER 3: MAXON MOTOR CONTROL**

**3.1 4-Q-DC Servo amplifier ADS 50/10**

**OPERATING INSTRUCTIONS**

The ADS 50/10 is a powerful servoamplifier for driving permanent magnet DC motors from 80Watts upto 500Watts. Four modes can be selected by DIP switches on the board:

* Speed control using tacho signals
* Speed control using encoder signals
* IxR compensated speed control
* Torque or current control

The ADS 50/10 is protected against excess current, excess temperature and short circuit on the motor winding. With the FET power transistors incorporated in the servoamplifier, an efficiency of up to 95% is achieved. A built in motor choke combined with the high PWM frequency of 50kHz allows the connection of motors with a very low inductivity. In most cases an external choke can be omitted.

Thanks to the wide input power supply range of 12-50 VDC, the ADS 50/10 is very versatile and can be used with various power supplies. The aluminium housing makes installation simple, with terminal markings for easy connection.

**CHAPTER 4 : PMAC**

**PROGRAMMABLE MULTI-AXIS CONTROLLER (PMAC)**

**4.1 Introduction**

The PMAC PCI-Lite is a member of the PMAC family of boards optimized for interface to traditional servo drives with single analog inputs representing velocity or torque commands. Its software is capable of eight axes of control, although it can have only four channels of on-board axis interface circuitry.

The PMAC PCI-Lite is a full-sized PCI-bus expansion card. While capable of PCI bus communications, with or without the optional dual-ported RAM, it does not need to be inserted into a PCI expansion slot. Communication can be done through an RS-232 or RS-422 serial port. Standalone operation is possible.

**4.2 Board Configuration**

The base version of the PMAC PCI-Lite provides a 1-1/2-slot board with:

* 40 MHz DSP563xx CPU
* 128k x 24 zero-wait-state flash-backed SRAM
* 512k x 8 flash memory for firmware and user backup
* RS-232/422 serial interface, PCI bus interface
* Four channels axis interface circuitry, each including:
  + - 16-bit +/-10V analog output
    - 3-channel differential/single-ended encoder input
    - Four input flags, two output flags
    - Interface to external 16-bit serial ADC
    - Display, control panel, mixed I/O, direct I/O interface ports
    - Buffered expansion port
    - Clock crystal with +/-100 ppm accuracy
    - PID/notch/feed forward servo algorithms

**4.3 PMAC Features**

* Capable of commanding up to 8 axes of motion simultaneously
* 256 separate motion programs can be stored at once
* Uses features of BASIC-type high level languages
* Serial Port Interface/PC bus Interface
* PMAC is a full computer in its own right, capable of standalone operation

**Fig 4.1 PMAC**

****

**4.4 Mounting**

The PMAC can be mounted in one of two ways: in the PCI bus, or using standoffs.

* PCI bus: To mount in the PCI bus, simply insert the P1 card-edge connector into the PCI socket
* Standoffs: At each of the four corners of the PMAC board, there are mounting holes that can be used to mount the board on standoffs.

**4.5 Power Supplies**

## 4.5.1 Digital Power Supply

2A @ +5V (+/-5%) (10W) (Eight-channel configuration with a typical load of encoders). The host computer provides the 5V-power supply if PMAC is installed in its internal bus. With the board plugged into the bus, it will pull +5V power from the bus automatically and it cannot be disconnected. In this case, there must be no external +5V supply, or the two supplies will fight each other, possibly causing damage. This voltage could be measured between pins 1 and 3 of the terminal block. In a stand-alone configuration, when PMAC is not plugged in a computer bus, it will need an external 5V supply to power its digital circuits.

## 4.5.2 Analog Power Supply

0.3A @ +12 to +15V (4.5W)

0.25A @ -12 to -15V (3.8W)

The analog output circuitry on PMAC is optically isolated from the digital computation circuitry, and so requires a separate power supply. Typically, this supply can come from the servo amplifier; many commercial amplifiers provide such a supply, or an external supply may be used.

# 4.6 Over Travel Limits and Home Switches

When assigned for the dedicated uses, these signals provide important safety and accuracy functions. While normally closed-to-ground switches are required for the over travel limits inputs, the home switches could be either normally closed or normally open types. However, for the following reasons, the same type of switches used for over travel limits is recommended:

* Normally closed switches are proven to have greater electrical noise rejection than normally open types.
* Using the same type of switches for every input flag simplifies maintenance stock and replacements.

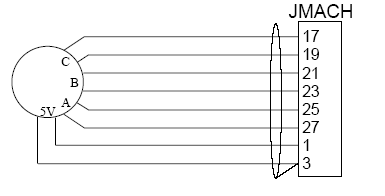
# 4.7 Motor Signals Connections (JMACH Connectors)

**4.7.1 Incremental Encoder Connection**

Each JMACH connector provides two +5V outputs and two logic grounds for powering encoders and other devices. The +5V outputs are on pins 1 and 2; the grounds are on pins 3 and 4. The encoder signal pins are grouped by number: all those numbered 1 (CHA1, CHA1/, CHB1, CHC1, etc.) belong to encoder #1. Usually, the encoder number matches the motor number, but it is not necessary. If the PMAC is not plugged into a bus and drawing its +5V and GND from the bus, these pins are used to bring in +5V and GND from the power supply.

**Example:** differential quadrature encoder connected to channel #1:

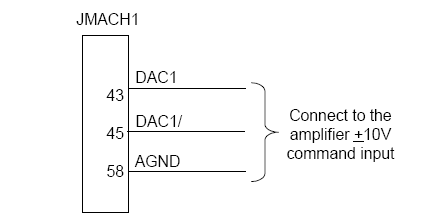
**Fig 4.2 Encoder connected to channel#1**



**4.7.2 DAC Output Signals**

If PMAC is not performing the commutation for the motor, only one analog output channel is required to command the motor. This output channel can be either single-ended or differential, depending on what the amplifier is expecting. For a single-ended command using PMAC channel 1, connect DAC1 (pin 43) to the command input on the amplifier. Connect the amplifier's command signal return line to PMAC's AGND line (pin 58). In this setup, leave the DAC1/ pin floating. Do not ground it. For a differential command using PMAC channel 1, connect DAC1 (pin 43) to the plus command input on the amplifier. Connect DAC1/ (pin 45) to the minus-command input on the amplifier. PMAC's AGND should still be connected to the amplifier common.

**Fig 4.3 DAC Output Signals- Connections**



**4.7.3 General-Purpose Digital Inputs and Outputs (JOPTO Port)**

PMAC’s J5 or JOPTO connector provides eight general-purpose digital inputs and eight general-purpose digital outputs. Each input and each output has its own corresponding ground pin in the opposite row.

Characteristics of the JOPTO port on the PMAC:

* 16 I/O points. 100mA per channel, up to 24V.
* Hardware selectable between sinking and sourcing in groups of eight; default is all sinking (inputs can be changed simply by moving a jumper; sourcing outputs must be special-ordered or field configured).
* Eight inputs, eight outputs only; no changes. Parallel (fast) communications to PMAC CPU.

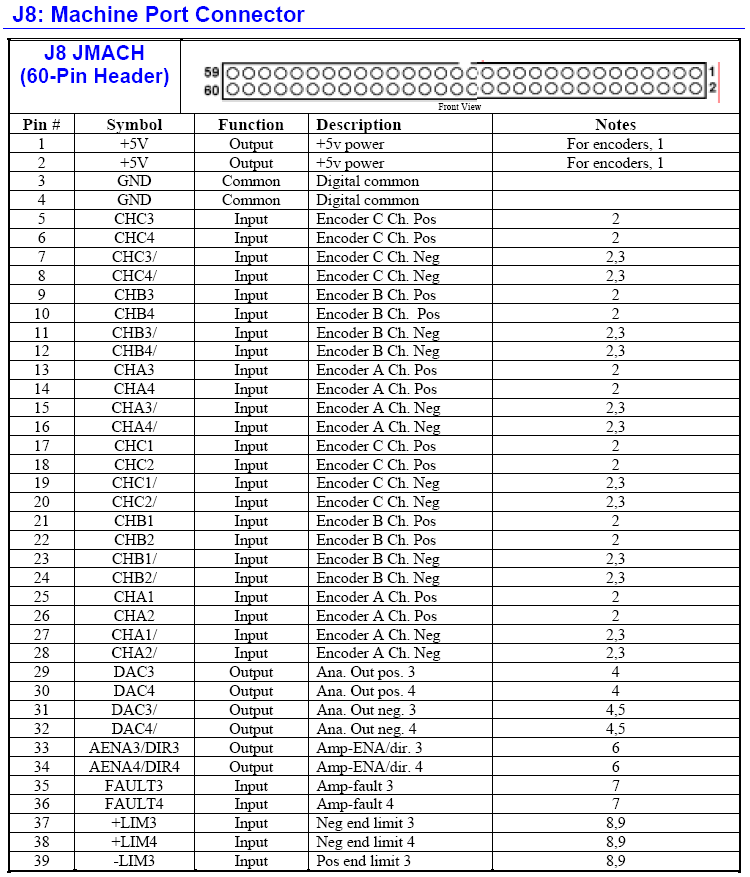
# 4.7.4 Control-Panel Port I/O (JPAN Port)

The J2 (JPAN) connector is a 26-pin connector with dedicated control inputs, dedicated indicator outputs, an encoder input, and a analog input. The control inputs are low true with internal pull-up resistors. They have predefined functions unless the control-panel-disable I-variable (I2) has been set to 1.

**4.7.5 Selector Inputs**

The four low-true BCD-coded input lines FDP0/ (LSBit), FDP1/, FDP2/, and FDP3/ (MSBit) form a low true BCD-coded nibble that selects the active motor and coordinate system (simultaneously). Usually, these are controlled from a single 4-bit motor/coordinate-system selector switch. The motor selected with these input lines will respond to the motor-specific inputs. It will also have its position following function turned on (Ix06 is set to 1 automatically.); the motor just de-selected has its position following function turned off (Ix06 is set to 0 automatically.).

Usually, these are controlled from a single 4-bit motor/coordinate-system selector switch. The motor selected with these input lines will respond to the motor-specific inputs. It will also have its position following function turned on (Ix06 is set to 1 automatically.); the motor just de-selected has its position following function turned off (Ix06 is set to 0 automatically.).



# 4.8 PMAC PCI-LITE Software Setup

**4.8.1 Communications**

With PEWIN32 Pro, a PMAC PCI-Lite board plugged in a PCI bus slot will be recognized by the operating system when the computer is booted up.

**4.9 PMAC Variables**

PMAC has a large set of Initialization parameters (I-variables) that determine the personality of the card for a specific application. Many of these are used to configure a motor properly. Once set up, these variables may be stored in non-volatile EPROM memory (using the SAVEcommand) so the card is always configured properly (PMAC loads the EPROM I-variable values into RAM on power-up).

**4.9.1 I-Variables (1024)**

* Initialization and setup variables
* They have pre-defined meanings
* Some affect whole card
* Some are motor specific
* Some are coordinate system specific
* Some are encoder specific

**4.9.2 P-Variables (1024)**

* General purpose user variables
* 48-bit floating point format
* Global access (regardless of the coordinate system)

**4.9.3 Q-Variables (1024)**

* General purpose user variables
* 48-bit floating point format
* Specific to a coordinate system

**4.9.4 M-Variables (1024)**

* Provide user access to memory and I/O
* User defines address, offset, and bit width
* Unsigned, 2’s compliment, BCD, floating point

Formats available

**4.10 Operational Frequency and Baud Rate Setup**

The operational frequency of the CPU is set in software by the PMAC I46 I- variable. If this variable is set to 0, PMAC firmware looks at the jumper E48 to set the operational frequency. If I46 is set to a value greater than 0, the operational frequency is set to 10MHz \* (I46 + 1), regardless of the jumper setting. If the desired operational frequency is higher than the maximum rated frequency for that CPU, the operational frequency will be reduced to the rated maximum. It is always possible to operate the Flex CPU board at a frequency below its rated maximum. I46 is used only at power-up/reset. To change the operational frequency, set a new value of I46, issue a SAVEcommand to store this value in non-volatile flash memory. Then issue a **$$$** command to reset the controller.

**4.11 Serial Addressing Card Number**

I0 controls the card number for software addressing purposes on a multi-drop serial communications cable. If I2 is set to 2, the PMAC must be addressed with the @n command where *n* matches the value of I0 on the board, before it will respond.

If the PMAC receives the @n command where *n* does not match I0 on the board, it will stop responding to commands on the serial port. No two boards on the same serial cable may have the same value of I0.

# 4.12 PEWIN32PRO

This software enables us to configure, control and trouble-shoot our PMAC (Programmable Multi-Axis Controller). At its core, Pewin32Pro provides a terminal, a text editor for editing Motion/PLC programs and a workspace environment.

Additionally, there is a suite of tools for configuring and working with PMAC and its accessories. Pewin32PRO is a development tool for creating and managing specific PMAC implementations.

**4.12.1 Features**

Pewin32Pro has:

* Multi-threading of Pewin32PRO real-time display windows.
* Enhanced graphing in PmacPlotPRO.

Pewin32PRO provides basic tools to configure, control and diagnose PMACs. Here is a partial list of Pewin32PRO’s features and capabilities:

* Workspace support
* Project management
* A terminal window

A real-time color editor with many options. Features include error tracking during downloads, color options for different commands and compatibility with standard C code.

**4.13 MENU OVERVIEW**

How Menus Work

Pewin32PRO uses a dynamic menu system. This means that the menu at the top of the screen changes content depending on what window has the current focus (is highlighted). The standard menu displayed when the terminal has focus looks like this:



However, the menu will change when, for example, the Watch Window is highlighted



If an option you expect to be available is not, make sure you have the proper window highlighted. If you have many windows open, use the Window menu item to select the one you are looking for.



There are very user-friendly context sensitive menus which pop up in many of the Pewin32PRO Windows. Use the right mouse button over a window to see the context sensitive menu.

****

**File Menu**

The File menu handles several things:

1. Opening an existing or creating a new text file.

2. Management of workspaces

3. Uploading and downloading PMAC related text files.

• **NEW FILE** – Opens the editor, if not already open, and creates a new text file ready for editing.

• **OPEN FILE** – Opens the editor, if not already open, and loads the selected file.

• **UPLOAD MOTION PROGRAM** – Uploads the specified PMC program into an editor window.

• **NEW WORKSPACE** – First saves the current workspace, provided the General Setup and Options has been set to do so (**See Setup** | **General Setup and Options**), closes all open windows, and asks you for the name of your new workspace. Pewin32PRO will load the last workspace opened at the time of exiting the application.

• **OPEN WORKSPACE** – Closes the current workspace, if any is open, and opens the selected workspace file.

• **SAVE WORKSPACE** – Saves the workspace whenever this menu item is selected.

• **SAVE WORKSPACE AS** – Renames the active workspace, but first saves the active workspace provided the General Setup and Options has been set to do so (**See Setup** | **General Setup and Options**)

• **CLOSE WORKSPACE** – Closes all open windows, and saves the workspace, provided the General Setup and Options has been set to do so (**See Setup** | **General Setup and Options**).

• **SHOW PROJECT MANAGER** – Hides or shows the Project Manager.

• **UPLOAD PROGRAM(S)** – Uploads the listed motion and PLC programs into a file and then open that file in the editor window.

• **UPLOAD VARIABLES** – Allows uploading a range of I, P, Q or M variables into an editor window.

• **DOWNLOAD FILE** – Allows downloading any file with valid PMAC commands.

• **EXIT** – Closes the program.

**4.13.1 Configure Menu**

The **CONFIGURE** menu lets you view and change current variable definitions and PMAC feature parameters.

In some of the options that allow you to change a value or definition, the change is sent to PMAC immediately after the value is changed. This allows you to verify automatically that the change in the input field has also occurred in PMAC. It also protects against faulty entries, since out of range numbers will not be accepted.

• **I VARIABLES** – There are two interfaces for listing and setting I variables; by category, or by numerical order. In addition, a special interface for MACRO I-Variables exists

• **P VARIABLES** – Allows setting P variable values.

• **Q VARIABLES** – Allows setting Q variable values.

• **M VARIABLES** – Allows setting M variable definitions and values.

• **COORDINATE SYSTEMS** – This menu choice enables you to alter the currently defined coordinate systems, or define new ones.

• **ENCODER CONVERSION TABLE** – This menu choice allows you to alter, update, save or retrieve the entries of the encoder conversion table.

• **ENCODER CONVERSION TABLE MACRO -** This menu allows you to alter, update, save or retrieve the entries of the encoder conversion table of a MACO station.

**4.13.2 View Menu**

The **VIEW** menu contains many interactive status displays as diagnostic tools. Many of the displays are updated in real-time.

• **TERMINAL** – Opens a terminal window.

• **WATCH WINDOW** – Opens a watch display window. You can have multiple watch windows open.

• **POSITION** – opens a position display window. You can have multiple position windows open.

• **CONNECTOR STATUS** – Allows monitoring the status of PMAC’s connectors.

• **MOTOR STATUS** – Displays the interpretation of status bits associated with a specified motor in real-time.

• **COORDINATE SYSTEM STATUS** – Displays the status of the specified coordinate system in real-time.

• **GLOBAL STATUS** – Displays the interpretation of the global status bits in real-time.

• **MOTOR SETUP SUMMARY** – Displays the configuration of a specified motor.

• **PROGRAM/PLC STATUS (AND UPLOAD)** – Displays the motion program and PLC program numbers, starting address and size of all “programs” in PMAC’s memory.

• **PLCC PROGRAM STATUS** – (Compiled PLC) displays the PLCC number, starting address and size of all compiled PLCs in PMAC’s memory.

# Backup Menu

The options in this menu allow you to save or restore complete or partial portions of PMAC configuration to disk.

• **UPLOAD CONFIGURATION** – Allows saving all or part of PMAC’s configuration to a disk file.

• **RESTORE CONFIGURATION** – Allows restoring all or part of PMAC’s configuration from disk.

• **VERIFY CONFIGURATION** – Verifies a specified configuration file.

**4.13.3 Setup Menu**

This pull down menu allows you to customize Pewin32PRO.

• **GENERAL SETUP AND OPTIONS -** Allows the customizing of Pewin32PRO's features.

• **FORCE ALL WINDOWS TO DEVICE NUMBER –** allows the user to switch all windows in PMAC

Executive to communicate with a specific device number. This device number is immediately tagged to

all of the closed windows as well through the workspace file.

• **SHOW MESSAGE WINDOW** – Displays the Message window.

• **SHOW PROJECT MANAGER** – Displays the Project Manager window.

**4.13.4 Tools Menu**

This menu item allows the launching of application tools that complement the Pewin32Pro.

**PEWIN32 PRO Software Reference Manual**

**10 Menu Overview**

• **PMAC TEST PRO -** Launches PmacTest program. PmacTest is now installed with Pewin32PROExecutive.

• **PMAC PLOT PRO** - Launches PmacPlot program (if it is installed).

• **PMAC TUNING PRO** - Launches Pmac Tuning program (if it is installed).

• **P1 SETUP AND TUNING PRO** - Launches the PMAC1 setup program (if it is installed).

• **P2 SETUP AND TUNING PRO** - Launches the PMAC2 setup program (if it is installed).

• **TURBO/UMAC SETUP PRO -** Launches the Turbo Setup program (if it is installed).

• **UMAC CONFIG PRO -** Launches the UMAC Configuration Setup program (if it is installed).

• **CUSTOMIZE TOOLS MENU –** Allows adding third party applications in the tool menu. These applications then can be launched from the tools menu. These applications are listed after the customize tools menu.

**4.13.5 Window Menu**

This menu is for managing the position and arrangement of any windows currently displayed. It is dynamically updated to allow easy selection of a specific window. Further, it shows all of PMAC Executive windows and on selecting a window bring it to the front.

**4.13.6 Help Menu**

The **HELP** menu options allow you to retrieve on-line information about PMAC, the Executive program, and the various help functions. You also have access to the two diagnostic routines provided by Pewin32PRO.

• **CONTENTS** – Displays this manual's content in the help file format.

• **WHY AM I NOT MOVING ?** – A diagnostic routine to help determine why a motor is not responding.

• **WHY IS MY PROGRAM NOT RUNNING ?** – A diagnostic routine to help determine why a program is not running.

• **ABOUT** – displays information about your version of Pewin32PRO, including the version number, the copyright, legal and licensing notices.

**4.13.7 Terminal**

The Terminal represents a direct connection to a PMAC. There are two parts to the Terminal, the **Entry** and **Response** windows. Any characters you type at the keyboard are sent to PMAC after pressing *<ENTER>.* Any characters that are sent from PMAC to the PC are displayed on the response window in a color corresponding to the current communications mode. If any command is rejected by PMAC, an error code is shown as well as its description (and possible remedies) displayed in red text (assuming I6 is set to 1, the default). You should always read this text, as it may affect your application.

The entry window keeps track of all unique commands sent to PMAC. To retrieve a previously sent command, press the down arrow to view to the bottom right of the window, select and press *<Enter>*.

**4.13.8 Position Window**

The position window displays motor position information. You can select the window to display position, velocity, or following error separately or in combination for all eight motors simultaneously. Position can be displayed for all motors.

**4.13.9 Position Menu**

When the Position window is highlighted, the menu changes to:

**POSITION** – Allows displaying/hiding position. Position display can be combined with velocity and/or following error displays.

**VELOCITY** – Allows displaying/hiding velocity. Velocity display can be combined with position and/or following error displays.

**FOLLOWING ERROR** – Allows displaying/hiding following error. Following error display can be combined with position and/or velocity displays.

**Units and Scales:**

**Position and velocity units** Error! Bookmark not defined. – Allows setting scaling and unit parameters for each motor. It also has the option to display/hide any motor.

To change the scale factor for displayed information, enter the appropriate number of encoder counts per user units (i.e. if you want to have 10,000 CTS = 1 inch, enter 10000). Next, specify a name for your user units (i.e. “inch”, “deg”, “rev”).

Select your velocity units (i.e. "per ms", “per second” or “per minute”). Enter in a value for optional rollover **Error! Bookmark not defined.** (i.e. 360 if your user units are degrees). A value of 0 for rollover indicates no rollover. Lastly, specify how many decimal places to the right of the decimal point you wish to have displayed (for inches, you may want to use a value of 3 so inches are displayed as “2.002 inches”). These parameters affect only the way information is displayed in the position window. They have no connection to PMAC itself.

Deselecting the *display* check box associated with a motor will cause the position window to not display information about that motor. This allows systems to display/hide any given motor.

**RESTORE DEFAULT VALUES** - Reverts all settings to Pewin32PRO’s default settings, including display mode, Cts/Unit, Dec. places, Rollover and units for Position and Velocity.

**SELECT FONT | SELECT COLOR** - Allows changing the appearance of the Position window.

**NEXT SET OF MOTORS** - Allows display of any one set of the motors for Turbo PMACs. By pressing the *<PgUp>, <PgDn>* or the *<UpArrow>, <DownArrow>* keys it is possible to change to the next set of motors in that order.

**Motor Status**

This display shows the interpretation of the status bits of the specified motor in real-time. This is done by continually inquiring the PMAC for corresponding X and Y registers of each motor. The corresponding registers are also displayed for each motor. True conditions are highlighted. The *<PgUp>, <PgDn>, <UpArrow>,* and *<DownArrow>* keys allow you to select which motor to examine. This status screen is equivalent to the “??” command at the terminal screen.

**MOTOR STATUS**



As of now, only full mode of display is provided. In future releases a custom status screen will be available for the user to get any status bit of a motor or coordinate system in the same screen. This status screen is equivalent to the “?” command at the terminal screen.

**4.14 Coordinate Systems Status:**

This display shows the status of the specified coordinate system in the real-time fashion. This is done by continually inquiring the PMAC for corresponding X and Y registers of each coordinate system.

Corresponding registers are also displayed for each coordinate system. True conditions are highlighted.



Pressing the <PgUp>or <PgDn>or the <UpArrow>,<DownArrow>keys changes the coordinate system being examined.

As of now, only full mode of display is available. In future releases a custom status screen will be available for the user to get any status bit of a motor or coordinate system in the same screen. Unlike Non-Turbo PMACs’ coordinate system status (48-bit wide), Turbo PMACs need to be polled for three registers (72-bit wide) in order to get the status of each coordinate system. This status screen is equivalent to the “??” command at the terminal screen.

**4.15 Coordinate Systems:**

This menu choice enables you to alter the currently defined coordinate systems, or define new ones. For motors to move within motion programs, they must first be assigned to an axis within one (and only one) of the eight (sixteen for Turbo) possible coordinate systems. Any motor may be assigned to any valid axis (X, Y, Z, A, B, C, U, V, or W) or coordinate system, provided the motor has not been previously assigned to another axis or defined in another coordinate system. To move from one coordinate system to the next, use the <*PgUp>* and <*PgDn>* keys. Assignments are downloaded to PMAC's internal memory as you ADD and REMOVE motors from the current axis .



**Coordinate System to Modify-Monitor:** This field displays the number of the coordinate system within PMAC that is currently being viewed. Use the **+** and **-** buttons or <*PgUp>* and *<PgDn>* to step through the coordinate systems.

**Current Axis Definitions:** This window lists all motors and their corresponding axis definitions for the current coordinate system.

**Edit:** This button opens a dialog box where you can edit the axis definition selected in the current axis definition window. If you want to undefine a motor definition, use the Remove button.

**Remove:** This button removes the motor selected in the current axis definition window from the coordinate system. Once you remove a motor from the current axis definition, it will appear in the Available Motors window, and is available again for axis assignment.

**Available Motors:** This window lists all the PMAC motors that are available to be assigned to a coordinate system. A motor is available if it is not already defined in any coordinate system.

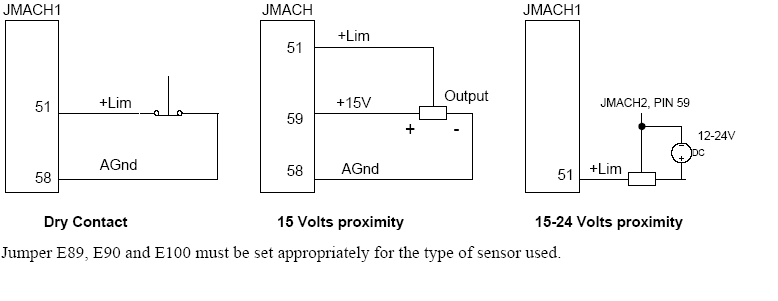
**Add:** This button adds the motor selected in the Available Motors window to the current axis definition window. When you press this button, a dialog box opens where you can enter the axis definition for the motor before it is added to the coordinate system.

**View all Coordinate Systems:** This option opens a window containing a list of all PMAC coordinate systems and their axis definitions. You cannot edit the coordinate system definitions in this window, but you can print this information.

**Done:** This closes the Configure Coordinate Systems window. For Turbo PMAC systems, a message box may pop up telling you the optimal setting for I68 (Maximum Coordinate System).

**4.16 Types of Over travel Limits**

PMAC expects a closed-to-ground connection for the limits to be considered not on fault. This arrangement provides a failsafe condition and therefore it cannot be reconfigured differently in PMAC. Usually a passive normally closed switch is used. If a proximity switch is needed instead, use a 15V normally closed to ground NPN sinking type sensor.



# 4.18 Applications and Uses of PMAC

PMAC can serve in wide variety of applications, from those requiring sub-micro inch precision to those needing power of hundreds of kilowatts or horsepower. Its diverse uses include:

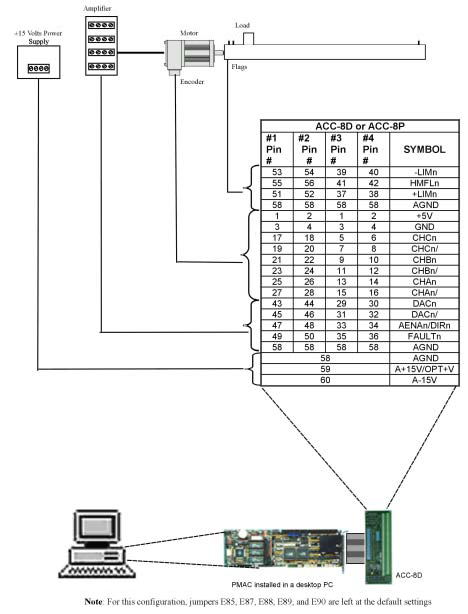
Robotics Assembly lines

Food processing Material handling

Machine tools Camera control

Printing paper and Silicon wafer processing

Lumber processing Automatic welding



#### Interfacing of the PMAC to the DC Servo Drive in closed loop

**DESCRIPTION**

The interfacing of the Drive to the PMAC can be explained with the block diagram shown above. It consists of an servo-drive amplifier (maxon motor) , PMAC chip, Servo Motor. The PMAC chip is installed in the computer. The position of the servomotor can be controlled by writing a program. According to the given commands and values, the PMAC software interfaces to the servo-amplifier drive. According to the given inputs and the commands given by the PMAC, the drive will generate a command signal, which is given to the axis drive. These signals are tracked initially using NI SCOPE SOFTWARE .This axis drive helps in positioning the servomotor by a command signal. The shaft of the axis drive can be positioned to specific angular positions by sending the servo command signal to it. As long as the coded signal exists on the input line, the drive will maintain the angular position of the shaft. As the command signal changes, the angular position of the shaft changes. The feedback provided by the axis drive is a positional feedback, which helps in controlling the position of the servomotor or the manipulator as shown.

###### CHAPTER 5 : TUNING

**5.1 Introduction**

The basic reasons for using servo systems in contrast to open loop systems include the need to improve transient response times, reduce the steady state errors and reduce the sensitivity to load parameters.

Improving the transient response time generally means increasing the system bandwidth. Faster response times mean quicker settling allowing for higher machine throughput. Reducing the steady state errors relates to servo system accuracy. Finally, reducing the sensitivity to load parameters means the servo system can tolerate fluctuations in both input and output parameters.

Servo control in general can be broken into two fundamental classes of problems. The first class deals with command tracking. It addresses the question of how well does the actual motion follow what is being commanded. The typical commands in rotary motion control are position, velocity, acceleration and torque. For linear motion, force is used instead of torque.

The second general class of servo control addresses the disturbance rejection characteristics of the system. Disturbances can be anything from torque disturbances on the motor shaft to incorrect motor parameter estimations used in the feedforward control. The familiar “P.I .D.” (Proportional Integral and Derivative position loop) and "P.I .V." (Proportional position loop Integral and proportional Velocity loop) controls are used to combat these types of problems.

**5.2 P.I .D. Control**

The basic components of a typical servo motion system are depicted in the figure below using standard Laplace notation. In this figure, the servo drive closes a current loop and is modeled simply as a linear transfer function G(s*)* (Of course the servo drive will have peak current limits, so this linear model is not entirely accurate, however it does provide a reasonable representation for our analysis). In their most basic form, servo drives receive a voltage command that represents a desired motor current. Motor shaft torque, Tis related to motor current, Iby the torque constant, Kt. The following equation shows this relationship. **T= Kt I**

**Fig 5.1 Basic PID Servo Control Topology**



The transfer function of the current regulator or really the torque regulator can be approximated as unity for the relatively lower motion frequencies we are interested in and therefore we make the following approximation shown.

**G(s)=1**

The servomotor is modeled as a lump inertia, J, a viscous damping term, b, and a torque constant, Kt. The lump inertia term is comprised of both the servomotor and load inertia. It is also assumed that the load is rigidly coupled such that the torsional rigidity moves the natural mechanical resonance point well out beyond the servo controller’s bandwidth. This assumption allows us to model the total system inertia as the sum of the motor and load inertia for the frequencies we can control. The actual motor position, θ(s) is usually measured by either an encoder or resolver coupled directly to the motor shaft. Again the underlying assumption is that the feedback device is rigidly mounted such that its mechanical resonant frequencies can be safely ignored. External shaft torque disturbances, Tdare added to the torque generated by the motor's current to give the torque available to accelerate the total inertia, J.

Around the servo drive and motor block is the servo controller that closes the position loop. A basic servo controller generally contains both a trajectory generator and a P.I .D. controller. The trajectory generator typically provides only position set point commands labeled in Fig as θ\* (s).

The P.I .D. controller operates on the position error and outputs a torque command that is sometimes scaled by an estimate of the motor's torque constant Kt. If the motor's torque constant is not known, the P.I .D. gains are simply re-scaled accordingly.

There are three gains to adjust in the P.I .D. controller, Kp*,* Ki and Kd. These gains all act to reduce the position error defined as:

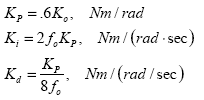
Error = θ\* (t) - θ (t)

**5.3 Tuning the P.I .D. Loop**

This involves selecting the values of gains Kp*,* Ki and Kd. Ziegler and Nichols [1] proposed an analytical for this purpose. Their procedure basically involves two steps:

**Step 1**: Set Ki and Kdto zero. Excite the system with a step command. Slowly increase Kpuntil the shaft position begins to oscillate. At this point, record the value of Kpand set Koequal to this value. Record the oscillation frequency, fo.

**Step 2**: Set the final P.I .D. gains using equation

****

The proportional term affects the overall response of the system to a position error. The integral term is needed to force the steady state position error to zero for a constant position command and the derivative term is needed to provide a damping action, as the response becomes oscillatory. Unfortunately all three parameters are inter-related so that adjusting one parameter will effect any of previous parameter adjustments.

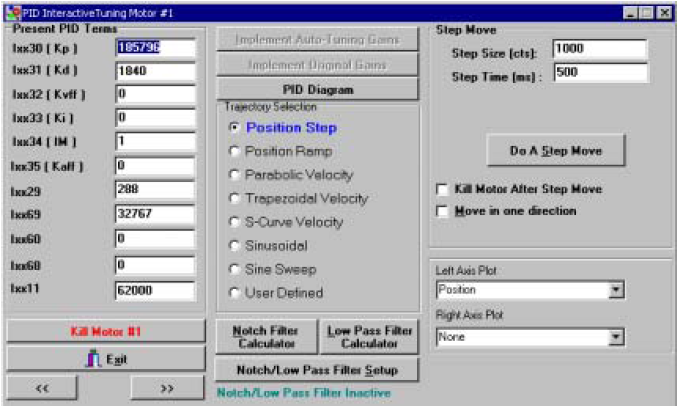
**5.4 Tuning using PMAC Software PRO**

Manual as well as automatic PID tuning can be done using this software.

**5.4.1 Regular PID Interactive Tuning**

This should ideally be used only for fine-tuning, after automatic tuning generates the “ball park figure” for required gain settings. In the following section simplified explanation of each gain and parameters is given.

**Fig 5.2 Tuning Window**



**5.4.2 Present PID Terms:**

* **Ixx30 (Kp) - Proportional Gain -** Increasing proportional gain stiffens the servo loop and increases the natural frequency of the closed loop system. Theoretically, increasing the proportional gain will result in improved positioning and tracking. However, often for real systems increasing the proportional gain increases their sensitivity to the noise and disturbances.
* **Ixx31 (Kd) - Derivative Gain -** Derivative gain works like damper. Higher the derivative gain, higher the damping action. This gain prevents overshoot but makes the system sluggish
* **Ixx32 (Kvff) - Velocity FF (feed forward) Gain** - Velocity feed forward gain will help the system with steady state error reduction. Setting it to an unreasonable value will destabilize the system. Often the optimal result is obtained by setting this value equal to the derivative gain value.

♦ **Ixx33 (Ki) - Integral Gain** - Integral gain acts to correct the system according to the accumulated following error of the system. It is particularly effective to counter the steady state error caused by friction. However, for numerical reasons and servo stability concerns, an excessively high value of the integral gain is discouraged. Begin tuning with a lower value and observe the improvements. If you want no integral action, set it to 0.

* **Ixx60 - Servo Cycle Extension** - If the your load is moving very slowly, that is, if the encoder counts relative to the servo cycle is low all the time, it may be a good idea to extend the servo cycle. By increasing the servo cycle extension, PMAC will try to close the loops less frequently. This is desirable, since PMAC will use less time for servo calculation. More calculation power would be reserved for PLCs and other “house keeping” chores.

* **Ixx68 - Friction FF (feed forward) Gain** - This gain acts when the position of the servo is not within ‘in position’ limits at zero velocity state (steady state), according to the direction needed to be compensated

**5.5 Position Step (Feedback Tuning with Step Response)**

Step response is often used as a method of evaluating a feedback filter. There are three key step response parameters to set the feedback. They are:

**Rise Time** — The time it takes the system to go from 10% to 90% of the commanded step. Natural frequency is directly related to this.

**Overshoot** — The percentage past the commanded step that the system travels. Damping ratio is directly related to this.

**Settling time** — The time it takes the system to get and stay within 5% of the commanded step.

**5.5.1 Doing the Step Response**

* Set a safe starting filter with a little proportional gain, with no derivative or integral gain, and no feedforward. The current values for Kp, Ki, and Kd are displayed on the screen.
* Do a step move and observe the plotted response displayed on the screen.
* Adjust (probably increase) Kp to get the fastest rise time possible without a huge amount of overshoot.
* Once you have a fast response, increase Kd to bring down the overshoot to the desired value. Note that this will also increase the rise time. You may need to do further tradeoffs between Kp and Kd to get the desired response.
* Once you have set Kp and Kd, you have taken care of your dynamic step response (provided you are using error integration only in position). Now you will want to add integration to improve the static holding properties of the system. As you increase Ki and observe the step response, you will notice that it increases overshoot but comes back to the command position more quickly. A good value for Ki is one that brings the response back down to the command position as quickly as possible without going back past it.

****5.6 Feedback Tuning with Step Response****

**Step response is often used as a method of evaluating a feedback filter. Many controls textbooks contain information on interpreting step responses for establishing proper feedback, particularly for second-order systems (Current-controlled motors driving inertial loads are second-order systems.) In a step response, a sudden change is made to the command position and the feedback filter attempts to bring the system to this new position. In observing how the system gets to the new position, we can deduce a great deal about the properties of the system. It does not matter that such a large instantaneous step in position in the actual operation of the system will ever be created. The purpose of this jolt to the system is to bring out system characteristics that might otherwise not be obvious. This detailed information on the PID filter is not essential to performing the tuning, but is included here for reference.**

**PMAC has three feedback parameters to be adjusted in this process:**

****1. Kp: Proportional gain (Ix30)****

****2. Kd: Derivative gain (Ix31)****

****3. Ki: Integral gain (Ix33)****

**Some systems will have mechanical resonances in the coupling of the motor to the load. The PID filter cannot compensate for these resonances; if their presence is not tolerable, keep the gains low enough not to stimulate them, or (preferably) stiffen the coupling to reduce the resonances.**

* **Kp = Proportional gain**
* **KI = Integral gain**
* **Kd = Derivative gain**

**First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal.**

**The signal (u) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error.**

http://www.engin.umich.edu/group/ctm/PID/PIDtuteq1.GIF

**This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again. This process goes on and on.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CL RESPONSE** | **RISE TIME** | **OVERSHOOT** | **SETTLING TIME** |  |
| **Kp** | **Decrease** | **Increase** | **Small Change** |  |
| **Ki** | **Decrease** | **Increase** | **Increase** |  |
| **Kd** | **Small Change** | **Decrease** | **Decrease** |  |

**An example of tuning for a step response using PMAC software PRO is given below:**



****Modified Step Response with Higher Kd (Ix31)**** ****Modified Step Response with Increased Kd****

**The next figure below shows the response with Kp increased from 80000 to 1000000. Note that the shape of the curve has not changed much (this is because the effective derivative gain is increasing with Kp), but the rise time has improved slightly (to 13 msec). We turn now to Kd.**

**The figure below shows the step response with a Kd of 2500. This is the critically damped case; i.e. fast response with no overshoot. With Kd any smaller, we get some overshoot. And with it any larger, we just slow down the response. The tendency of the system to settle slightly off from the target position is due to a net torque or static friction. We will eliminate this with integral gain.**



****Modified Step Response with Increased Ki (Ix33)****

**The figure below shows what happens with a little bit (relatively speaking) of integral gain (Ki = 10000): the steady- state error is gone, but the nature of the curve has not really changed.**



****Modified Step Response with High Ki****

****The figure below** shows the response curve for a substantially increased Ki (100000). This curve demonstrates how quickly the system would respond to a disturbance while in position. (Remember that we are using integral gain only when in position, so changing integral gain does not affect the actual dynamic response, although it will change the shape of the step response here. We still have the stability characteristics of critical damping while on the move.) Even the higher value of Ki does not result in oscillations, so we will use that. We have achieved what we wanted with feedback.**

