The Unimation Puma servo system

MTM-226

Peter I. Corke
July 1994

CSIRO Division of Manufacturing Technology
Locked bag 9, Preston,
AUSTRALIA. 3072
pic@mlb.dmt.csiro.au

CSIRO
AUSTRALIA

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This report describes in some detail the servo system of the Unimation Puma robot, a common laboratory and industrial robot. The Unimation controller has a classical hierarchical structure as shown in Figure 1.1. The host, in the factory configuration an LSI-11/2 running the VAL robot language, communicates with the Arm Interface Board (AIB) over a bidirectional parallel bus.

The AIB in turn communicates with six digital servo boards over a custom wired DEC double-height backplane, see Figure 1.2. These boards execute commands (such as position setpoint setting, reading current encoder values, and miscellaneous parameter setting) as well as implementing the position control loop. They connect through the backplane to the analog servo boards which implement nested velocity and current control loops. The older Mark I controller uses two boards per axis: one digital and one analog. In newer Mark II controllers,

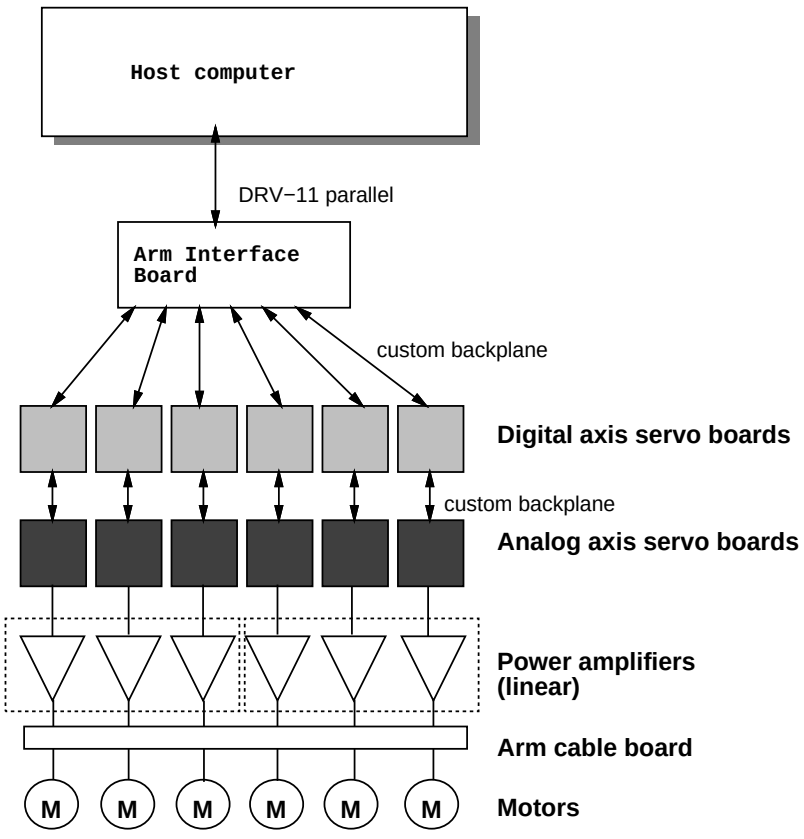

Figure 1.1: Structure of Unimation Puma Mark I servo system.

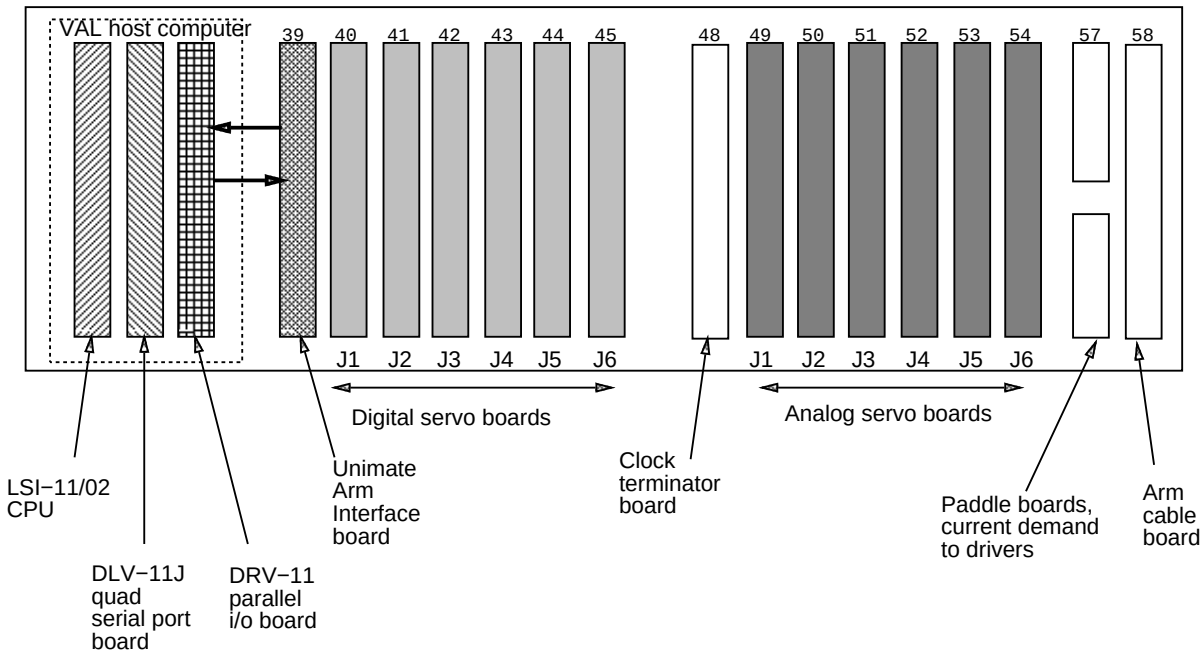

Figure 1.2: Controller backplane card layout for Mark I controller (viewed from front of backplane).

the digital and analog boards have been combined into a single servo board per axis.

The AIB connects directly to a DEC DRV-11A parallel port board, which in the factory configuration is driven by the LSI-11/2. However any parallel port conforming to the cabling and electrical conventions of the DRV-11A may be connected to the AIB, and this is the basis of many 'foreign' controllers in research laboratories that use Unimation Puma robots [13, 21].

Many details, particularly those relating to the firmware functionality, vary greatly between machines. Details here relate to firmware revision 6, which corresponds with VAL-I revision 12, release circa 1981.

Chapter 2 describes the operation of the digital servo boards, including hardware and firmware. Chapter 3 describes the electromechanical axis dynamics, including friction, inertia, current loop, velocity loop and position loop. The appendices provide additional information on topics such as robot parameters, joint communications, and encoder calibration.

1.1 Nomenclature

Electrical signals prefixed by a slash '/' are active low; the notation used on Unimation schematics is an 'L' suffix to the signal. ICs on circuit board are prefixed by an upper-case 'U' followed by a letter and a digit which designate the coordinate of the IC on the board. Circuit diagrams for these two servo boards may be found in the Unimate Technical Manual [20]. The most commonly used symbols and their units are listed in Table 1.1. The following conventions have also been adopted:

- Time domain variables are in lower case, frequency domain in upper case.
- Transfer functions will frequently be written using the notation

$$K(a)[\zeta, \omega_n] = K\left(\frac{s}{a} + 1\right)\left[\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1\right]$$

Symbol	Description	Unit
$\underline{v}$	a vector	
$\underline{v}_x$	a component of a vector	
$\mathbf{A}$	a matrix	
$\hat{x}$	an estimate of x	
x_d	demanded value of x	
$\mathbf{A}^T$	transpose of A	
B	viscous friction	Nms/rad
f	force	N
G	gear ratio	
$\mathbf{G}$	gear ratio matrix	
i	current	A
j	$\sqrt{-1}$	
J	scalar inertia	kgms^{-2}
$\mathbf{J}$	inertia tensor, 3×3 matrix	kgms^{-2}
k, K	constant	
K_i	amplifier gain (transconductance)	A/V
K_m	motor torque constant	Nm/A
L	inductance	H
$\underline{q}$	generalized joint coordinates	
$\underline{Q}$	generalized joint torque/force	
R	resistance	Ω
θ	angle	rad
$\underline{\theta}$	vector of angles, generally robot joint angles	rad
s	Laplace transform operator	
$\underline{S}_i$	first moment of link i. $\underline{S}_i = m_i \underline{s}_i$	kgm
t	time	s
T	sample interval	s
τ	torque	Nm
τ_C	Coulomb friction torque	Nm
v	voltage	V
ω	frequency	rad/s
z	z-transform operator	

Table 1.1: Frequently used symbols

A free integrator is an exception, and it should be taken that $(0) = s$.

- When specifying motor motion, inertia and friction parameters it is important that a consistent reference is used, usually either the motor or the load, denoted by the subscripts m or l respectively.

  For numeric quantities the units radm and radl are used to indicate the reference frame.

Chapter 2

Digital control architecture

2.1 The Arm Interface Board (AIB)

The AIB is the interface between the host computer and the digital servo boards. It allows the host computer to send commands and read the status of individual digital servo boards via a simple parallel data interface. The signals to the host, shown in Figure 2.1, are:

- `Data in`: 16-bit command or data words to the servo system
- `Data out`: 16-bit data words from the servo system
- `CSR0`: controls whether the data input should be interpreted as command or data

$$\texttt{CSR0} = \begin{cases} 0 & \text{command} \\ 1 & \text{data} \end{cases}$$

- `REQA`: the rising edge signifies that the digital servo board has completed the last command.
- `REQB`: when true indicates that one or more digital servo boards are requesting 'attention'. The servo attention word can be read to determine which axis has requested attention.
- `NDR` (New data ready): a strobe pulse (approximately 1 μs wide) from the host indicating that new data is available on the `data in` lines. In command mode (`CSR0` = 0), this causes the command word to be latched and the appropriate servo select line (`SEL` bus) driven low. In data mode (`CSR0` = 1) latches it causes `XFER_REQ` to be asserted to the digital servos.
- `DTRANS` (Data transferred): a strobe pulse (approximately 1 μs wide) from the host indicating that it has completed reading the data on the `data out` lines.

Note that the strobe pulses should be around 1 μs long. If the strobe is still high when the digital servo board responds, REQA will not rise. Example communications code is given in Appendix C.

2.1.1 Commands

A digital servo command is an 8-bit value, written to the AIB, with `CSR0` = 0, as shown in Figure 2.2. The low order three bits of the command word are decoded and used to address a particular joint via the `SEL` bus. The next four bits specify the command to be executed by the digital servo board. Commands may:

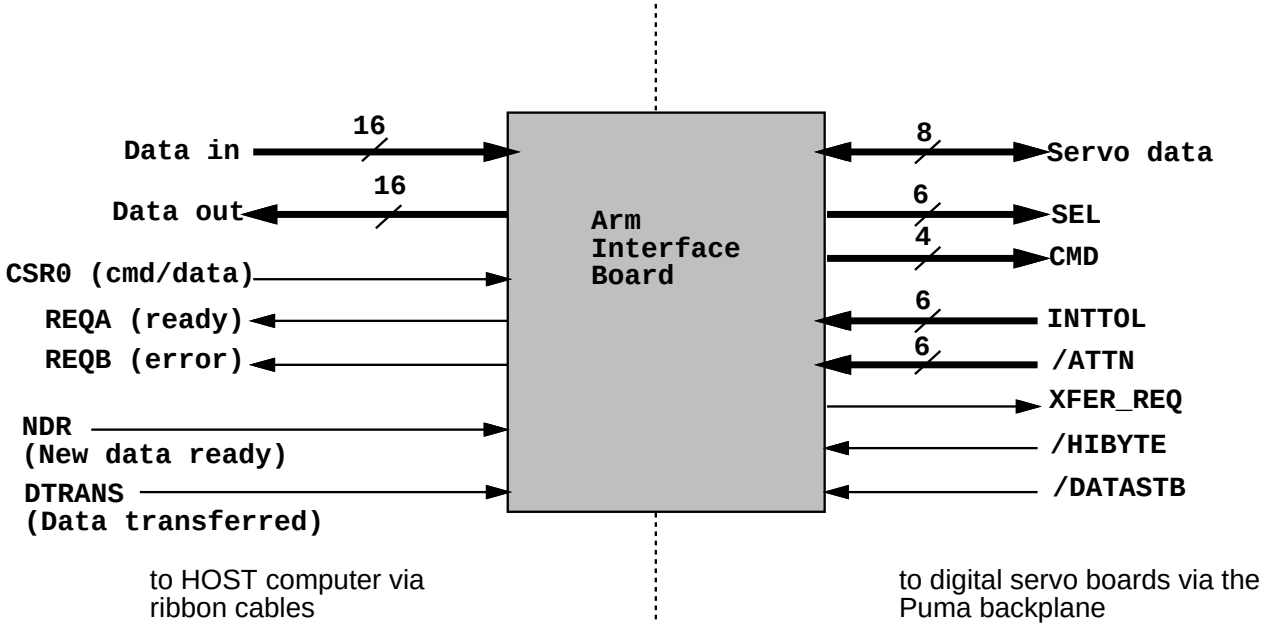

Figure 2.1: Signals to and from the arm interface board.

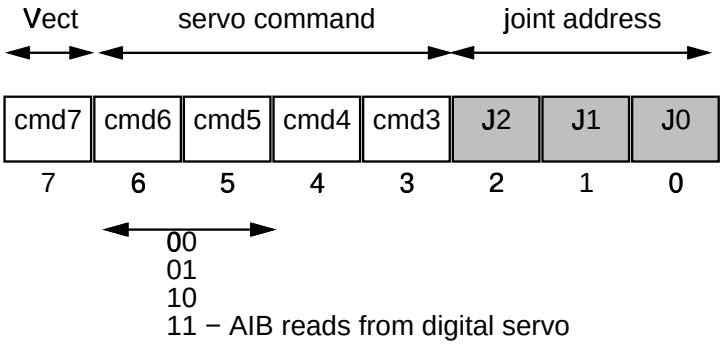

Figure 2.2: Host command word format, written with CSR0 = 0.

- write data to the servo, but the command is not executed until a data word is given (NDR strobe);

- read data from the servo where the command is executed immediately and the result is written by the digital servo board to the AIB upon completion and from where it is read by the host.

If the high order bit of the command is set the interface operates in 'vector' mode and the joints are accessed sequentially. This is useful when the same operation, read or write, needs to be performed for all joints. The command word need be written only once prior to the sequence of data transfer operations. When a vector mode command is received the joint address is latched in counter U6B and incremented after each command completion, thus addressing the digital servo boards in sequence. Figures 2.4 and 2.5 show the details.

The direction of data transfer between the AIB and the digital servo board is implicit in the command. If bits 5 and 6 are both equal to 1 the AIB reads from the servo board, otherwise the command is a write.

CSR0	SELECT7	Read	Write
0 (CMD)	0	Tolerance and attention bits[1]	Digital servo command word
1 (DATA)	0	Digital servo data word	Digital servo data word
1 (DATA)	1	WX + status[2]	OX + hand status[3]

Note 1. The eight tolerance bits appear in the low byte of the input word, and the joint attention bits in the high byte. A joint attention bit (active low) is reset by a digital control board when it encounters an error, or a zero index. Generally the joint status of the signaling axis should be read to determine the exact problem.

Note 2. When reading WX the data word is:

Bit	Purpose
0-7	WX external input lines
8	Arm power off
9	STOP (same as teach box OFF)
10	RUN (from the front panel switch)
11	RESTART (from the front panel switch)

Note 3. When writing OX, the data word is:

Bit	Purpose
0-7	OX external output lines
8	High power enable which allows arm power to be enabled with the front panel button
9	Hand open
10	Hand close

If neither hand open or hand closed are asserted the hand is in a 'relaxed' state.

Table 2.1: Details of SELECT7 mode operation.

2.1.2 SELECT7 mode operation

If the command word's joint address is 7 the AIB operates in SELECT7 mode and the host communicates with the AIB, not a digital servo board. Reading data in SELECT7 mode does not require waiting for `REQA` to become true. This mode gives access to plant input (WX) and output (OX) signals, front panel switches, joint tolerance and attention lines. The various operating modes are given in Table 2.1.

2.1.3 Major data paths

The major data paths are shown in Figure 2.3. The command byte is latched by U6D and the low order three bits are decoded to drive the servo select lines. Data direction over the digital servo bus is implicit in the command type: commands that read from the digital servo have bits 6 and 7 of the command word set. This is used by U2B to control the direction of the servo bus transceivers.

The axis processors read and write 16-bit host data via the 8-bit servo data bus. Data written by the host is latched by U4D/U5D which the axis processor can read via the servo data bus. The active axis processor controls which byte is placed by the AIB onto the servo bus by means of `/HIBYTE`. A command that reads data from the axis processor must wait until the axis processor has written data into the 16-bit register (U4D, U5D) on the AIB. The axis processor places 8-bits of data at a time onto the servo bus, and uses `/HIBYTE` to select which half of the AIB register will be written, and `/DATASTB` to latch it.

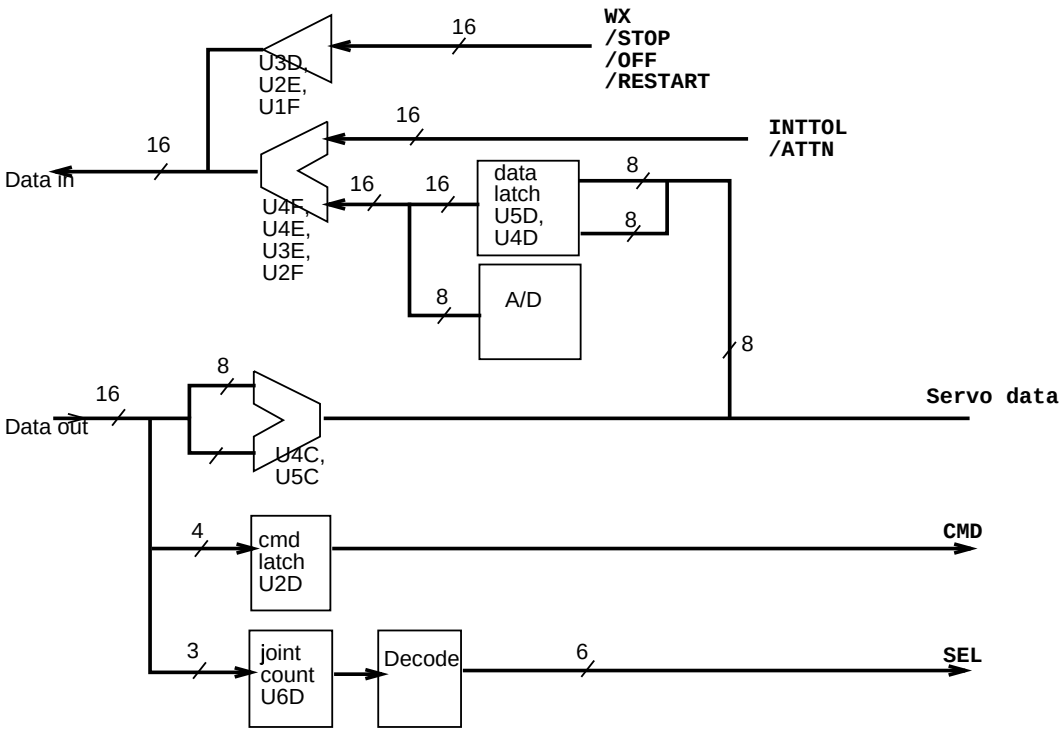

Figure 2.3: Major data paths in the AIB.

The READADC command is a special case since no data is read from the digital servo board. Instead the output of the ADC on the AIB is gated onto the output data bus. The READADC command is processed by the digital servo boards, but they simply acknowledge the command after a programmed delay, to cover the conversion time of the ADC (around 100 μs) and provide the `REQA` acknowledgment.

2.1.4 The digital servo backplane

The Puma backplane, shown in Figure 2.1, carries a number of signal busses between the digital servo cards and the arm interface board. Some busses are common to all boards (such as `servo data`), while others have one signal connected to each board, via the custom backplane (such as `SEL`). The busses include:

- `CONTROL` bus. This bus contains the data coordinating signals:
  - `/HIBYTE` used by the servo board to select which byte the AIB places on the servo data bus;
  - `/DATASTB` data strobe, used by the servo board to latch data into the AIB;
  - `/XFER_REQ` strobe indicating valid command on `CMD` bus, and causes an NMI to the selected axis processor;
  - `HALT` halt operation of the axis processor;

  Data transfer direction is implicit in the current command. `/XFER_ACK` is asserted by the joint servo when it has completed a command, and toggles the state of the AIB's `REQA` line to signal the host. In 'vector' mode, it is also used to increment the joint counter on the arm interface card (U6D) via the `/CTUP` signal.

- `SEL` bus. Each board is selected by one of six joint select lines in the `SEL` bus. These lines are driven by a demultiplexer, U6E, on the arm interface card from the lower 3-bits (joint

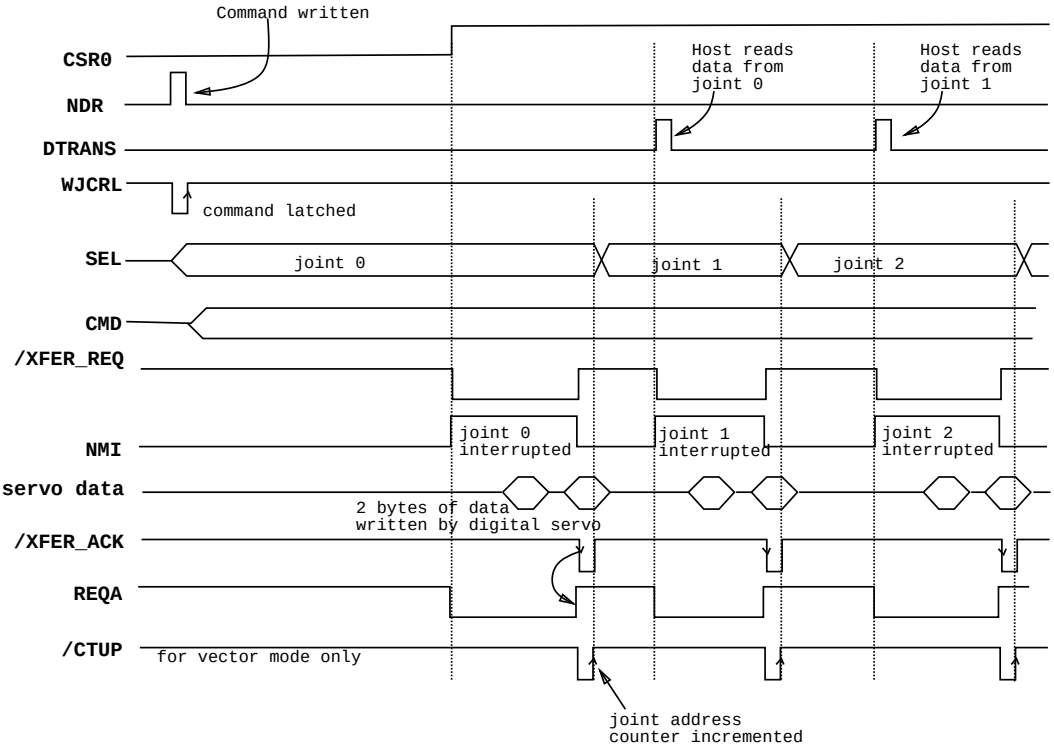

Figure 2.4: Timing details for vector mode reading.

number field) of the servo command word. When the card is selected and `/XFER_REQ` is asserted the axis processor is interrupted.

- `CMD` bus. This bus is provided in parallel to all servo cards and is acted upon only by the selected card. The command consists of bits 4 to 7 of the servo command word.
- `Servo data` bus. This bidirectional 8-bit bus is used to transfer 16-bits of data between the joint servo cards and the arm interface card. For servo card writes, signal `/HIBYTE` indicates which byte the arm interface must latch and `/DATASTB` latches the data. For reading, `/HIBYTE` selects which byte the arm interface must multiplex onto the data bus.
- `INTTOL` bus. This bus has one line per servo card. Each board asserts its line (active high) when the in-tolerance condition is true. This is usually when the encoders indicate the joint is within a specified distance from the destination (settable by the SETINT command).
- `ATTN` bus. This bus is similar to the `INTTOL` bus. A board asserts its `ATTN` line (active low) when it has an error condition, or has detected a zero index when in hunt zero index mode. The logical OR of all `ATTN` lines is available as the `REQB` signal of the arm interface.

Figures 2.4 and 2.5 show details of the timing relationships involved in reading and writing to the axis processors.

2.1.5 Analog to digital converter

The AIB contains an ADC0816 16 channel 8-bit ADC for digitizing joint potentiometer signals. More modern Mark 1 controllers, including those built by Kawasaki, utilize a 10-bit converter. The ADC output is tri-stated onto the internal data bus when it is read, but only the low order 8 or 10-bits are significant. The converter has a 100 μs conversion time.

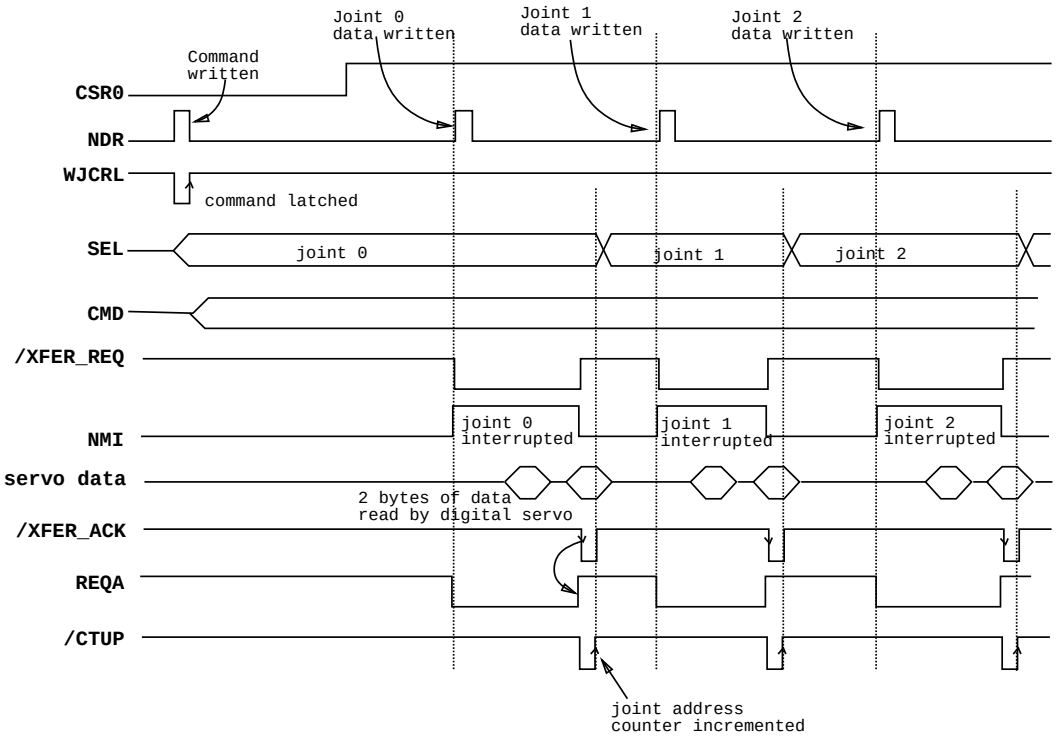

Figure 2.5: Timing details for vector mode writing.

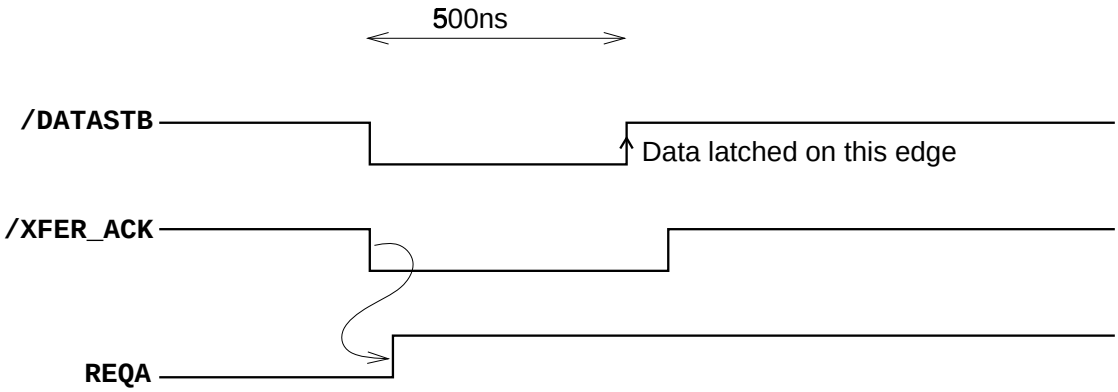

Figure 2.6: Timing details of `REQA` handshake line. Note delay between `REQA` being asserted, and the second data byte being latched.

When a READADC command is issued, the command is decoded by the AIB (U1B) which initiates conversion from the channel selected by the low order bits of the command word. The command is also given to the digital servo card which simply delays for a time before acknowledging. When the host reads the data word it takes the output of the ADC not the content of the data latch.

2.1.6 Communications bus timing

For a host reading data from the AIB there is a potential timing problem as shown in Figure 2.6. The second data byte is not in fact latched into the AIB until 500 ns after the `REQA` acknowledge signal is given. This problem was not observed with early, slow, host computers. A fast host should introduce some delay after the rising edge of `REQA` before assuming that the data word

has been completely latched.

We have measured that REQA rises around 75 μs after the command has been issued. In that time the 6503 has been interrupted, fetched the command, executed the appropriate handler and read or written 2 bytes of host data. Examination of the code reveals a number of critical sections during which interrupts are masked out and this could conceivably increase the axis response time.

Under some circumstances a rising edge on REQA may be caused for other reasons. If the previous command was an AIB (SELECT7) operation REQA will be left in the low state. REQA is set when a digital servo board asserts XFER_REQ. It is cleared on (U3A, U2B and U6A)

$$\texttt{RESET} + \texttt{CSR0}(\texttt{NDR} + \texttt{DTRANS.VECMODE})$$

Thus an AIB operation will leave REQA deasserted; the data transfer will clear it, and since no digital servo board is involved, it will not be set at completion of the operation.

2.2 Digital Servo Card Operation

2.2.1 The hardware

The digital servo board uses a 6503, an 8-bit microprocessor clocked at 1 MHz (from a clock line on the servo backplane) to close the axis position loop at approximately 1 kHz and to execute commands from the host, see Table 2.2. A 2716 EPROM contains up to 2kbytes of firmware and a 6532 multifunction chip provides 128 bytes of RAM, a counter/timer, and some parallel input/output capability. The memory map for the 6503 based servo card is shown in Figure 2.7. Other I/O facilities, detailed in Table 2.4, include:

- the AIB servo data bus for reading from and writing to the AIB;
- the host command bus;
- current, and index latched, encoder values.

and are shown in Table 2.4.

2.2.2 Interrupts

The 6503 axis processor receives the following interrupts:

- **NMI** whenever the axis processor is selected (SEL bus line asserted) and there is a valid command on the CMD bus (/XFER_REQ asserted). NMI may be masked out via a control register bit during critical code sections.
- **IRQ** from the 6532 due to timer countdowns (clock tick), and the zero index signal via input PA7 of the 6532 device (U4A).

2.2.3 Control register (CR)

The control register is an 8-bit output port, U2B, whose bits are described in Table 2.3. The 6503 firmware can control the operation of the analog servo board by means of:

- /INT to enable integral action on the analog servo board;
- /CURR_MODE to enable current mode where the DAC output commands motor current rather than motor velocity;

- /LSB_SERVO_MODE to enable high-gain position servo mode[1];
- /SERVO_ENABLE to enable the analog velocity loop

communicate joint status information to the host:

- TOL if the axis is within the specified position tolerance;
- ATTN if the axis requests attention

and mask out NMI interrupts during critical sections of code

- NMI_ENABLE.

2.2.4 Read multiplexer

The read multiplexer consists of devices U1B-1E on the joint digital servo card. It multiplexes four 8-bit data sources onto the 6503's data bus:

- Encoder up/down counter (U2C/D)
- Encoder counter latch (U3C)
- Current value of CR
- Host CMD bus

2.2.5 Encoder interface

The quadrature encoder signals are buffered and shaped on the analog servo board and passed via the backplane to the digital servo board. A state machine (U5D and associated logic) generate a count and direction signal from the encoder data. The logic generates 4 pulses per encoder line, so an n line encoder generates $4n$ pulses per revolution. An 8-bit up/down

[1] This signal is not driven by the current 6503 firmware, and the analog servo board does not use the output of the LSB servo amplifier.

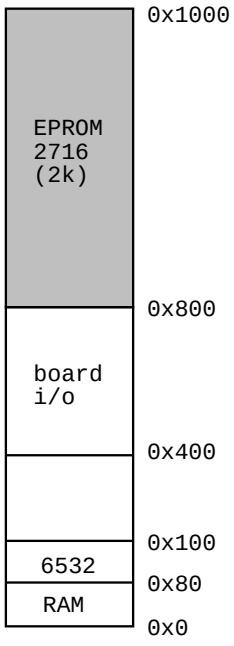

Figure 2.7: Axis processor memory map.

Command	Value	Read/Write	Description
POSMODE	0	W	Command motion in position control mode
CURMODE	0x08	W	Command current directly (no position control)
SPOSTL	0x10	W	Set position tolerance, set the width of the band around the target point within which the axis will assert in tolerance signal.
SETPOS	0x18	W	Set position encounter value
CALIB	0x20	W	Calibrate
SETDC	0x28	W	Set DC offset
SETINT	0x30	W	Set integration band tolerance
STDATA	0x38	W	Store data to 6503 RAM
STOPMDE	0x40	W	Enter stop mode
NOP	0x48	W	Do-nothing
NOP	0x50	W	Do-nothing
NOP	0x58	W	Do-nothing
READPOS	0x60	R	Read position counter
READSTAT	0x68	R	Read 6503 joint status word
READADC	0x70	R	Read ADC (joint potentiometer)
DIAGREAD	0x78	R	Read from the 6503 RAM location pointed to by memory location 2 (set up by a previous STDATA operation).

Table 2.2: Axis processor commands.

Bit	Value	Meaning
0	01	NMI_ENABLE
1	02	ATTN
2	04	/SERVO_ENABLE
3	08	/INT
4	10	/LSB_SERVO_ENABLE
5	20	/CURRENT_MODE
6	40	
7	80	TOL

Table 2.3: Control register bit definitions.

counter (U2C/D) counts the encoder edges and its output is available to the 6503 via the read-multiplexer. The counter has a range of only 256 encoder edges so every clock interval (924 μs) the hardware counter is read and a 24-bit software counter updated. On encoder index (once per motor revolution) the output of the up/down counter is latched (U3C) and the 6503 interrupted. This latched output is available to the 6503 via the read-multiplexer.

The RAM location `encmax` contains the number of encoder counts between zero indices and is used by the firmware. It is an 8-bit value which is interpreted as being offset 0x300. The usual values are 0xe8 (0x3e8 = 1000) or 0x20 (0x320 = 800). The Kawasaki firmware seems not to require this constant.

2.2.6 Digital to analog converter

A DAC80 12-bit DAC is connected to the parallel output lines of the 6532 device. The bipolar output voltage is in the range -10 V to 9.995 V. This voltage is used to command the analog

Name	Address	Operations	Usage
busl	400	rw	low byte of host data bus
bush	401	rw	high byte of host data bus
buslak	402	rw	lo byte of host bus, assert /XFER_ACK
bushak	403	rw	hi byte of host bus, assert /XFER_ACK
ctl	600	rw	counter lo
wad	601	w	apparently unused signal
	601	r	encoder value latched at zero index
cr	602	rw	control register (CR)
	603	r	host bus command/status byte

Table 2.4: Digital servo board I/O addresses.

velocity loop in position control mode, or the current loop in current control mode. The voltage is buffered by a unity-gain op-amp stage on the digital board with feedback coming from the analog board. This is perhaps to counter the effect of voltage loss due to poor contacts between the two servo boards.

2.2.7 AIB data I/O

Four locations are used by the 6503 to read/write data to the AIB. The locations `busl` or `bush` read or write data to the servo data bus and cause generation of the appropriate `/HIBYTE` and `/DATASTB` signals. Reading or writing to the locations `buslak` or `bushak` has the same effect, but also asserts the end of operation handshake `/XFER_ACK` and is used for the second (last) byte transferred.

2.3 Digital servo board firmware

The firmware revision level for our Mark I controller is 5.0, and the following sections describe the algorithms used. The code is written in a 'C' like pseudo-code and covers control strategy, interrupt front end, and host command handling. The commented disassembly, in a form for reassembly can be obtained by anonymous ftp from `asgard.mlb.dmt.csiro.au` in the file `pub/pic/servo.s.Z`.

2.3.1 Position control strategy

This function, summarized in Figure 2.8, is quite straightforward. However the implementation is complex since it is mostly done with triple precision, fixed point arithmetic. The least significant byte is considered as being to the right of the binary point. It performs setpoint interpolation at the 6503 tick rate, as well as implementing the position control loop. It is called every 924 μs and updates the current setpoint e'_d by an amount e_Δ. This increment is computed each time the host provides a new setpoint but the setpoint itself is not actually kept, see Section 2.3.3. Each cycle the current arm position is read by `getenc()` and the result placed in a circular buffer, `pos`, with PBUF elements.

The position control algorithm implemented is a simple digital PD law

$$v_{DAC}(k) = \frac{10}{2048}\left\{(e'_d(k) - e_m(k)) + D\,(e_m(k) - e_m(k - \texttt{velgan}))\right\} \tag{2.1}$$

where e'_d is the desired encoder value from the interpolator, and e_m is the current measured encoder count. This control law has been verified by disassembly of the microprocessor's EPROM and by direct measurement of the transfer function V_{DAC}/E_m. The second term above is a

```
#define PBUF    16

pos_control()
{
        i = (i - 1) % PBUF;
        pos[i] = getenc();              /* put encoder value into pos buffer */

        if (CR & SERVO_ENABLE) {
                e_d += e_delta;    /* update target position */
                if (--count == 0)
                        e_delta = 0;    /* no movement if past count */
                err = pos[i] - e_d;
                /*
                 * set TOL and INT bits
                 */
                if (abs(err - postol))
                        TOL = 1;
                else
                        TOL = 0;
                if (abs(err - inttol))
                        INT = 1;
                else
                        INT = 0;

                /*
                 * PD control
                 */
                u = P * err + dcoffset;
                u >>= outscal;
                j = (i + velgan) % PBUF;
                deriv = pos[i] - pos[j]; /* compute deriv fb */
                u += deriv;
                dacout(u);

                pending();              /* handle pending commands */
        }
        else {
                enc = pbuf[i];
        }
}

new_setpoint(e_new)
{
        e_delta = (e_new - e_d) / ninter;
        count = ninter;
}
```

Figure 2.8: Position control algorithm.

simple derivative obtained by taking the difference between the current position in the circular buffer, and the `velgan`th one before that. The output is a velocity demand to the analog velocity loop which is described in more detail in Section 3.6.

The position tolerance (SPOSTL command) is checked every tick and used to drive the joint's TOL bus line, indicating to the host that the joint is close to its setpoint. Similarly the integration band (SETINT command) is checked, and used to enable integral action on the analog velocity loop when the joint is close to its target position.

2.3.2 Interrupt front-end

The mainline code, shown in Figure 2.9, is invoked by a hardware reset from the bus. Note that the axis processor is able to be reset by the host via the AIB.

```
main()
{
        for (;;) {
                {initialize stack}
                {initialize memory}

                {read encoders}
                init_pos_buffer();

                while (control_bus & HALT_BIT == 0)
                        ;

                while (control_bus & HALT_BIT)
                        ;
        }
}

nmi()
{
        {push registers}
        while (f = jtab1[--cmd])  /* while pending commands */
                (*f)(arg[cmd]);   /* call handler */
        {restore registers}
}

irq()
{
        {push registers}
        if ({6532 timer int}) {
                { clear interrupt request }
                { reload interval counter}
                pos_control();
        }
        else
                joint_index();
        {restore registers}
}
```

Figure 2.9: Interrupt and main-line code.

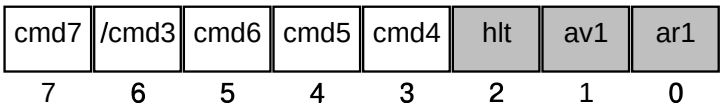

Figure 2.10: Transposed command bits as seen by the axis processor. The reason for this transposition is not known.

2.3.3 Host command handling

As mentioned above, when a command is written to a selected joint servo card an NMI is generated. NMIs are in fact controlled by external logic and a bit in the CR register, allowing them to be masked off during certain critical code sections. The NMI handler uses the `CMD` bus value as an index into a dispatch table whose contents are used as the low byte of a jump address `0x0e00 + table1[CMD]`. The command handlers are executed 39 cycles (39 μs) after the NMI is received. The format of the `CMD` bus value as it appears to the 6503 is not simple, as some of the bits have been transposed as shown in Figure 2.10. The handlers called fall into two categories:

1. Those for DIAGREAD, READSTATUS, READPOS, READADC and NOP are handled immediately, and the return values (if any) are written to the arm interface card. For READADC, a software timing loop delays the acknowledge to allow for the 100 μs ADC

conversion time.

2. Those for CURMODE, POSMODE, SETPOS, SETDC, STDATA, SPOSTL, CALIB, SETINT and STOPMDE are not handled immediately but are placed into a four deep circular buffer containing the command, and 16-bit parameter. They are executed by the `pending()` function at the end of each control update. CURMODE and POSMODE will perform a NOP if servo mode is disabled in the joint status word. The 6503 acknowledges these commands within approximately 75 μs.

The `pending()` function will service all pending commands. The saved commands are again used to find a handler address, but this time via a different dispatch table, and the complete jump address is `0x0d00 + table2[CMD]`. The commands may be grouped:

- SETPOS, SETDC, SPOSTL, SETINT, STDATA simply copy their parameter into the memory location associated with each variable.
- STOPMDE operates by setting the encoder demand e_d to the current position, $pos[i]$, effectively servoing to the current point, and disables current mode.
- CURMODE sign extends the data and outputs it to the DAC immediately. This means that current commands can be output every clock tick, and will be synchronized to the clock tick.
- POSMODE computes the encoder increment needed at each clock tick:

```
e_delta = (e_new - e_d) / ninter
```

Note that if the host is sending many commands to a single axis the command buffer may overflow. The axis acknowledges a command in 75 μs, but will execute them only every 924 μs. In that time 12 commands could potentially be sent and exceed the 4 buffers actually available.

2.3.4 Joint status word

The lowest two bytes of 6503 memory are known as the joint status word, and the READSTAT command is available to return that as a word value. The low byte is stored in location 0, the high byte in location 1. The joint status word is set using the STDATA command, one byte at a time. The significance of the bits are shown in Table 2.5 as per [21]. Our firmware appears not to implement some status flags such as overheated motor, DAC saturated timeout, out of position enveloped and lost synchronization with encoder. DAC saturation refers to the condition of maximum DAC output sustained for some period of time.

2.3.5 Timing

The 6532 multifunction device contains an 8-bit interval counter that decrements every 1, 8, 64 or 1024 Φ_2 edges. The counter can generate an interrupt when it decrements to zero. The counter has no initial value register so it cannot reload and continue counting.

In the joint servo card the Φ_2 rate is 1 MHz (from the backplane clock signal generated in the clock/terminator card). The software sets the interval counter's divisor to the value contained in location 8, initialized to 0x6f or 111, and the count mode to decrement every 8 Φ_2 edges. This corresponds to an interval of 896 μs,

$$interval = (divisor + 1) * factor$$

$$factor = 1,\ 8,\ 64\ or\ 1024$$

Bit	Value	Description
0	0x01	RAM error
1	0x02	No zero index
2	0x04	Out of position envelope
3	0x08	Lost synchronization with encoder
4	0x10	DAC saturated timeout
5	0x20	Overheated motor
6,7		Not used
8	0x80	Look for zero index (set by user). The joint responds to position demands as normal, but stops moving at the first zero index encountered, and the joint's attention line is asserted.
9	0x100	Set by the joint when a zero-index has been found. Further move commands will be ignored while this bit remains high.
10,11		Not used
12	0x1000	Divide-by-two mode
13	0x2000	Not used
14	0x4000	Enable integration inside band
15	0x8000	Enable servoing (joint will not move until this is set)

Table 2.5: Joint status word.

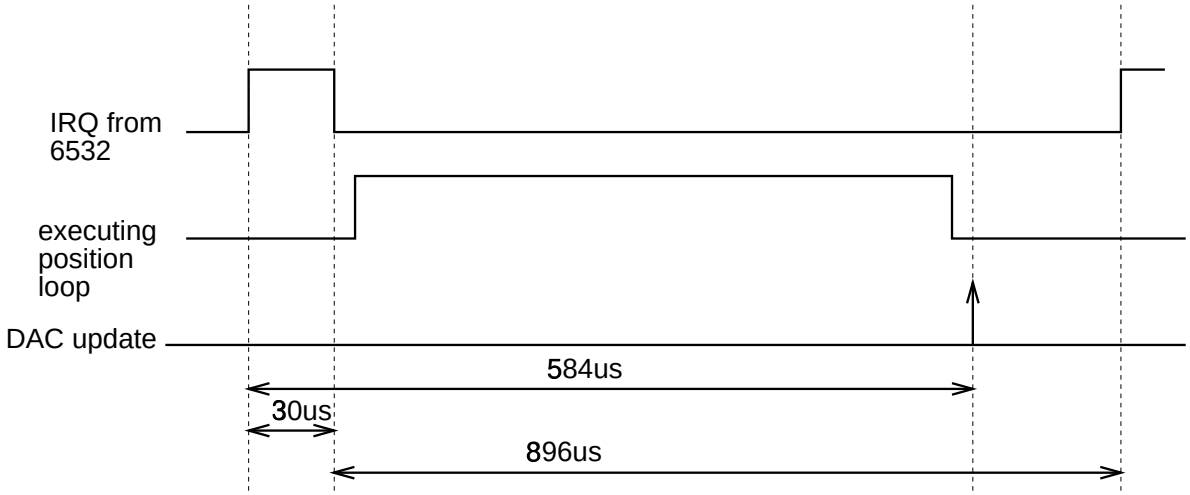

Figure 2.11: Timing details of axis processor firmware.

Measurements with a logic analyzer indicate that the 6503 takes 30 μs from IRQ being asserted by the 6532 to the software reseting the IRQ flag. The software then reloads the 6532 interval counter. Thus the measured interval between IRQ edges of 924 μs corresponds approximately to 896+30 μs. The DAC is updated some 584 μs after IRQ is asserted as shown in Figure 2.11.

2.3.6 Host synchronization

Two variables in the 6503 memory control the number of servo cycles between host setpoint commands and these are shown in Table 2.6. Note that if a setpoint command occurs later

interval	no.of ticks	`intscl`	`ninter`
224 ms	256	0	256
112 ms	128	1	128
56 ms	64	2	64
28 ms	32	3	32
14 ms	16	4	16
7 ms	8	5	8

Table 2.6: Setpoint timing control.

than expected the joint will stop by servoing to the last setpoint. If host setpoints come more slowly than expected the joints will stop before the next setpoint arrives, leading to very rough motion. If the command comes early the encoder increment is adjusted at the next tick and servoing will continue, but at a different velocity.

The number of ticks expected between host setpoint commands is kept in the variable `ninter`. Note that the tick period of 924 μs is not an integral divisor of the host setpoint update rates used. The encoder increment is scaled by `intscl` to suit the fixed point arithmetic used in the control algorithm. Since the scaling is done by shifting, not division, `ninter` is restricted to being a power of 2. Each time a POSMODE command is processed the variable `count` (shown in Figure 2.8) is reset to `ninter`.

The 1 MHz clock is divided on the clock/terminator card to periodically assert the non-vectored backplane interrupt line, BEVENT, which is used by VAL to initiate setpoint computation. The interval is adjusted by jumpers E1-6 on the clock/terminator card to one of the values shown in Table 2.6 — 28 ms is the rate used by VAL-I. 'Foreign' hosts do not have access to this interrupt signal, but the setpoint algorithm is such that it synchronizes with periodic host setpoint commands.

2.3.7 6503 RAM locations

Addresses of program variables within the address space of the 6503 have been identified from the disassembly and are shown in Table 2.7.

2.3.8 Discrepancies

There are some discrepancies between this document and the information in [21]. This may be due to different firmware revisions between the systems described.

The joint status word error bits for RAM error, out of position envelope and lost synchronization with encoder, do not appear to be implemented. Bits 6 and 7 which are described as not used, are in fact used internally by the encoder data handling routines.

The `velgan` constants for the Puma 500 and 600 series on [21] indicate that the `velgan` memory location is 0x5c which appears to be unused in our firmware. From the disassembly the correct address for this variable is 0x04.

2.3.9 Reassembly of axis processor firmware

The annotated source code has been reassembled using the public domain Frankenstein cross assembler, `as6502`. Other assemblers could be used but some differences in syntax and pseudo-ops could be expected.

The hex output of the assembler has been loaded and matches byte for byte with the data read from the original EPROM. Thus the facility exists to modify or enhance the joint servo firmware.

Address	Name	Initial value	Usage
00	`jstatus`	00	joint status (low byte)
01	`jstatus+1`	00	joint static (high byte)
02	`rmemp`		memory pointer for DIAGREAD command
03	`propgan`	01	Proportional gain term
04	`velgan`	00	Velocity (or derivative) gain
05	`outscal`	00	control force output scale factor
06	`ninter`	20	Number of clock ticks between setpoint commands
07	`intscl`	03	Scale factor related to `ninter`
08	`divisor`	6f	6532 clock divisor factor
09	`encmax`		Number of encoder counts per revolution
0c..0a	`e_delta`		Encoder increment at each tick
0f..0d	`e_target`		Encoder target for this tick
11..10	`dcoffset`		DC offset
14..12	`err`		Encoder error for this tick
17..15	`deriv`		Derivative for this tick
1a..18	`u`		Computed control force
1c..1b	`jvec1`	0d	Vector for jump table 1
1e..1d	`jvec2`	0e	Vector for jump table 2
1f	`i`		Position buffer pointer
3f..20	`pos`		Position buffer
41	`pending`	00	Number of commands pending
43	`cp`	00	Command buffer pointer
4f..44	`cbuf`		Command buffer
50	`count`		Tick number: $0 \leq$ `count` $\leq$ `ninter`
53..52	`postol`	0000	Position tolerance band
55..54	`inttol`	0000	Integral action tolerance band
58..56	`enc`		Encoder in fixed point form
7e	`stacktop`		system stack

Table 2.7: Axis processor firmware memory map.

Chapter 3

Axis electromechanical dynamics

This chapter details the electro-mechanical axis dynamics due to the robot's control electronics, actuators and mechanical transmission. These effects are at least as significant as the rigid-body effects which are the topic of much robotic 'dynamics' literature, but are less well covered perhaps due to their robot specific nature.

The model development in this chapter is based on experimental measurements and identification[1], as well as analysis of the robot's electrical schematics. The result is a detailed dynamical model which can be used for control design and simulation. Much of the material in this chapter is based on the axis model developed in Corke [5, 7]. Additional data on Puma robot parameters can be found in Corke and Armstrong-Hèlouvry [6, 7].

3.1 Friction

Dynamic effects due to the transmission or drive system are significant, particularly as the drive train becomes more complex. The addition of gears to amplify motor torque leads to increased viscous friction and non-linear effects such as backlash and increased Coulomb friction.

[1] Experiments were conducted primarily on joint 6. While this is typical of other axes on this, and other, Pumas those characteristics will depend on servo tuning and individual motor and axis characteristics.

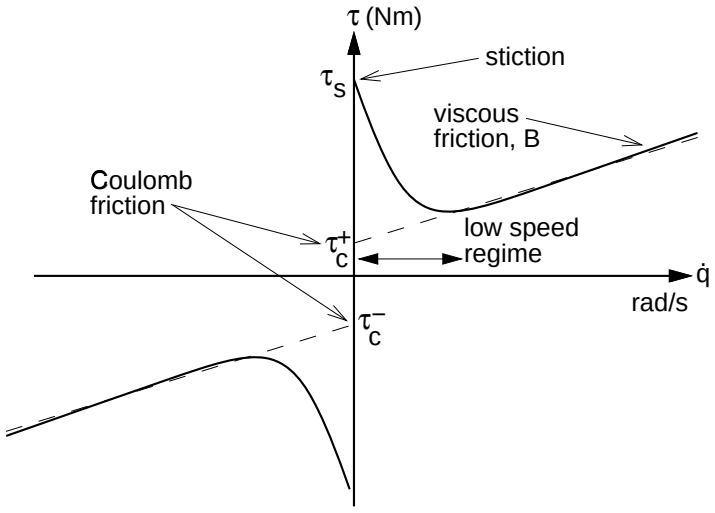

Figure 3.1: Typical friction versus speed characteristic. The dashed lines depict a simple piecewise-linear friction model characterized by slope (viscous friction) and intercept (Coulomb friction).

For a geared manipulator, such as the Puma, friction is a dominant dynamic characteristic. A typical friction torque versus speed characteristic is shown in Figure 3.1. The dashed line represents the simple friction model

$$\tau_f = B\dot{\theta} + \tau_c \tag{3.1}$$

where slope represents viscous friction, and offset represents Coulomb friction. The latter is frequently modeled by the non-linear function

$$\tau_c = \begin{cases} 0 & \text{if } \dot{q} = 0 \\ \tau_c^+ & \text{if } \dot{\theta} > 0 \\ \tau_c^- & \text{if } \dot{\theta} < 0 \end{cases} \tag{3.2}$$

and in general $|\tau_c^+| \neq |\tau_c^-|$. Static friction, or stiction, is the torque that is necessary to bring a stationary joint into motion, and can be considerably greater than the Coulomb friction value. The more complex model, represented by the solid line, is significant only for very low velocities [4]. The negative slope can be attributed to the Stribeck effect due to mixed lubrication — the load is partially, but increasingly, lifted by lubricant resulting in decreasing friction. When the contact is fully lubricated viscous friction is evident. This negative slope characteristic can result in instability with simple PID joint control schemes at low velocity.

The friction parameters discussed represent total lumped friction due to motor brushes, bearings and transmission. They are dependent upon many factors including temperature, state of lubrication, and to a small extent shaft angle. Armstrong [3,4] provides detailed investigations of the low speed and angular dependence of joint friction for a Puma 560 manipulator.

Classic techniques for determining friction are based on the simple piecewise-linear friction model of (3.1). A joint is moved at constant velocity and the average torque (typically determined from motor current) is measured. This is repeated for a range of velocities, both positive and negative, from which the slope (viscous friction coefficient) and intercepts (Coulomb friction torques) can be determined. Measurement of joint friction characteristics using this approach have been previously reported for a Puma 260 [12] and a Puma 560 [11].

The friction values for the robot used in this work have been determined experimentally and are summarized in Table 3.1. Coulomb friction and viscous friction were determined by measuring average joint current for various joint speeds over a short angular range about the vertical 'READY' position. This was done to eliminate the torque component due to gravity which would otherwise influence the experiment. A typical plot of current versus velocity is shown in Figure 3.2. Given knowledge of the motor torque constant from Table 3.3, viscous and Coulomb friction values may be determined from the slope and intercept respectively. A robot 'work out' program was run prior to the measurements being taken, so as to bring joints and lubricant up to 'typical' working temperature. There is no evidence of the negative slope on the friction curve at the velocities used here. The lowest velocity in each test was $5°/s$ at the link, which is approximately 5% and 2% of the peak velocities for the base and wrist joints respectively.

From Table 3.1 it is clear that some friction parameters show considerable dependence on the direction of rotation. Statistical analysis of the mean and variance of the sample points [22] for positive and negative velocity for each joint indicate that at the 95% confidence level the friction values are not equal, apart from viscous friction values for joints 1 and 6. Armstrong [4] showed statistically that Coulomb and viscous friction had separate values for positive and negative velocities. For linear system design and simulation the mean viscous friction value will be used.

Stiction, τ_s, was measured by increasing the joint current until joint motion occured[2]. For those joints subject to gravity load the robot was positioned so as to eliminate gravity torque.

[2] Taken as increasing encoder value for 5 consecutive sample intervals.

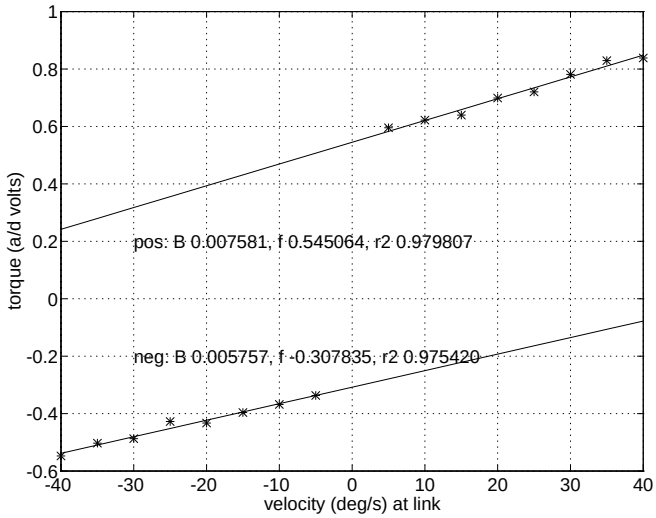

Figure 3.2: Measured motor current (actually motor shunt voltage) versus joint velocity for joint 2. Experimental points and lines of best fit are shown.

Joint	τ_s^+	τ_C^+	B^+	τ_s^-	τ_C^-	B^-	$\overline{B}$
1	0.569	0.435	1.46e-3	-0.588	-0.395	-1.49e-3	1.48e-3
2	0.141	0.126	0.928e-3	-95.1e-3	-70.9e-3	-0.705e-3	0.817e-3
3	0.164	0.105	1.78e-3	-0.158	-0.132	-0.972e-3	1.38e-3
4	14.7e-3	11.2e-3	64.4e-6	-21.8e-3	-16.9e-3	-77.9e-6	71.2e-6
5	5.72e-3	9.26e-3	93.4e-6	-13.1e-3	-14.5e-3	-71.8e-6	82.6e-6
6	5.44e-3	3.96e-3	40.3e-6	-9.21e-3	-10.5e-3	-33.1e-6	36.7e-6

Table 3.1: Measured friction parameters — motor referenced (Nm and Nms/rad). Positive and negative joint velocity are indicated by the superscripts. The column $\overline{B}$ is the mean of B^+ and B^-.

The average stiction over 20 trials was taken. The standard deviation was very high for joint 1, around 16% of the mean, compared to 5% of the mean for the wrist joints.

3.2 Motor

The Puma robot uses two different size of motor — one for the base axes (joints 1-3) and another for the wrist axes (joints 4-6). Data on these motors is difficult to find, and the motors themselves are unlabeled. There is speculation about the manufacturer and model in [1], and it is strongly rumored that the motors are 'specials' manufactured by Electrocraft for Unimation. It is conceivable that different types of motor have been used by Unimation over the years. Tarn and Bejczy et al. [17, 19] have published several papers on manipulator control based on the full dynamic model and cite sources of motor parameter data as Tarn et al. [18] and Goor [8]. The former has no attribution for motor parameter data quoted, while the latter quotes "manufacturer's specifications" for the base motors only. The source of Tarn's data for the wrist motors [19] is not given. Kawasaki manufacture the Puma 560 under licence, and data on the motors used in that machine was provided by the local distributor. Those motors are Tamagawa TN3053N for the base, and TN3052N for the wrist. However some of these parameters appear different to those quoted by Tarn and Goor.

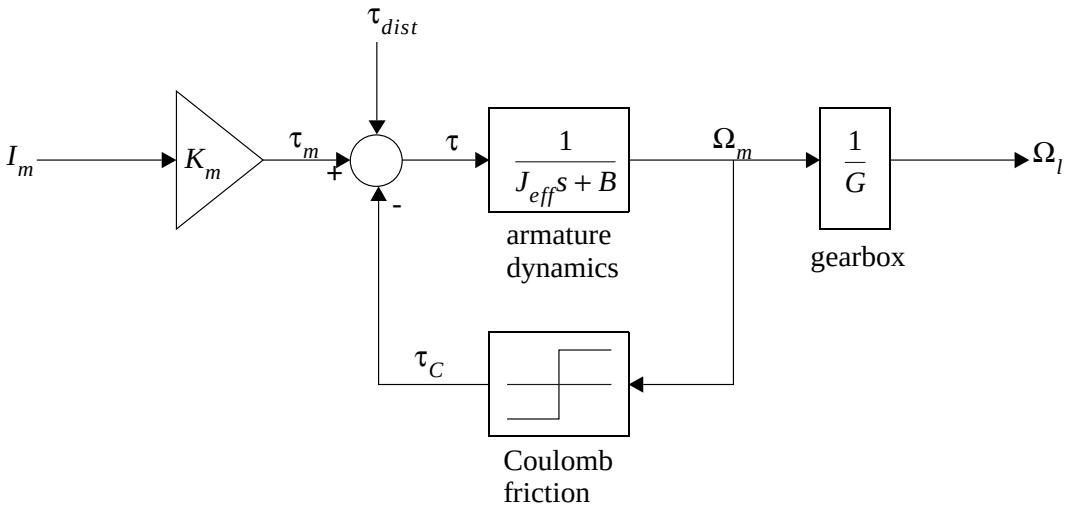

Figure 3.3: Block diagram of motor mechanical dynamics. τ_{dist} represents disturbance torque due to load forces or unmodeled dynamics.

A complete block diagram of the motor system dynamics is shown in Figure 3.3 and assumes a rigid transmission. The motor torque constant, K_m, is a gain that relates motor current to armature torque

$$\tau_m = K_m i_m \tag{3.3}$$

and is followed by a first-order stage representing the armature dynamics

$$\Omega_m = \frac{\tau}{J_{eff}s + B} \tag{3.4}$$

where Ω_m is motor velocity, J_{eff} the effective inertia due to the armature and link, and B the viscous friction due to motor and transmission. The so-called *mechanical pole* is given by

$$p_m = -\frac{B}{J_{eff}} \tag{3.5}$$

Coulomb friction, τ_c, described by (3.2), is a non-linear function of velocity that opposes the armature torque. The friction and inertia parameters are lumped values representing the motor itself and the transmission mechanism. Finally, there is a reduction gear to drive the manipulator link.

An equivalent circuit for the servo motor is given in Figure 3.4. This shows motor impedance comprising resistance, R_m, due to the armature winding and brushes, and inductance, L_m, due to the armature winding. R_s is the shunt resistor which provides the current feedback signal to the current loop. The electrical dynamics are embodied in the relationship for motor terminal voltage

$$V_m = sK_m\Theta + sL_mI_m + (R_s + R_m)I_m + E_c \tag{3.6}$$

which has components due to back EMF, inductance, resistance and *contact potential difference* respectively. The latter is a small constant voltage drop, typically around 1 to 1.5V [9], which will be ignored here. The so-called *electrical pole* is given by

$$p_e = -\frac{R_m}{L_m} \tag{3.7}$$

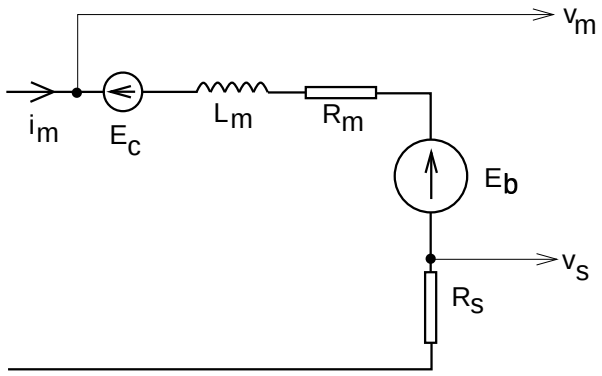

Figure 3.4: Schematic of motor electrical model.

Parameter	Armstrong	Tarn	Kawasaki	Preferred
J_{m1}	291e-6	198-6	200e-6	200e-6
J_{m2}	409e-6	203e-6	200e-6	200e-6
J_{m3}	299e-6	202e-6	200e-6	200e-6
J_{m4}	35e-6	18.3e-6	20e-6	33e-6
J_{m5}	35e-6	18.3e-6	20e-6	33e-6
J_{m6}	33e-6	18.3e-6	20e-6	33e-6

Table 3.2: Comparison of motor inertia values from several sources — motor referenced ($\mathrm{kg\,m^2}$).

3.2.1 Inertia

There are two components of inertia 'seen' by the motor. One is due to the the rigid-body link dynamics 'reflected' through the gear system, and the other due to the rotating armature. The total inertia sets the upper bound on acceleration, and also affects the location of the mechanical pole by (3.5).

Several sources of armature inertia data are compared in Table 3.2. Armstrong's [2] values (load referenced and including transmission inertia) were divided by G_i^2, from Table A.2, to give the values tabulated. These estimates are based on total inertia measured at the joint with estimated link inertia subtracted and is subject to greatest error where link inertia is high. From knowledge of motor similarity the value for motor 2 seems anomalous. Values given by Tarn [18] are based on an unknown source of data for armature inertia, but also include an estimate for the inertia of the shafts and gears of the transmission system for the base axes. These inertia contributions are generally less than 2% of the total and could practically be ignored. The very different estimates of armature inertia given in the literature may reflect different models of motor used in the robots concerned. The preferred values for the base axes are based on a consensus of manufacturer values rather than Armstrong, due to the clearly anomalous value of one of his base motor inertia estimates. Inertia of the drive shaft, flexible coupling and gear will be more significant for the wrist axes. Frequency response measurements in Section 3.4 are consistent with the higher values of Armstrong and therefore these are taken as the preferred values.

Change in link inertia with configuration has a significant effect on the dynamics of the axis control loop and creates a significant challenge for control design if it is to achieve stability and performance over the entire configuration space.

Parameter	Armstrong	Paul [16]	CSIRO
K_{m_1}	0.189	0.255	0.223
K_{m_2}	0.219	0.220	0.226
K_{m_3}	0.202	0.239	0.240
K_{m_4}	0.075	0.078	0.069
K_{m_5}	0.066	0.070	0.072
K_{m_6}	0.066	0.079	0.066

Table 3.3: Measured motor torque constants - motor referenced (Nm/A).

3.2.2 Torque constant

For a permanent magnet DC motor the torque and back EMF are given by [9]

$$\tau = \frac{Z}{2\pi}\phi\, i_m = K_m i_m \tag{3.8}$$

$$E_b = \frac{Z}{2\pi}\phi\dot{\theta} = K_m\dot{\theta} \tag{3.9}$$

where ϕ is the magnetic flux due to the field, Z the number of armature windings, and θ the motor shaft angle. If a consistent set of units is used, such as SI, then the torque constant in Nm/A and back-EMF constant in Vs/rad will have the same numerical value.

Armature reaction is the weakening of the flux density of the permanent magnet field, by the MMF (*magneto-motive force* measured in Ampère-turns) due to armature current. This could potentially cause the torque constant to decrease as armature current is increased. However according to Kenjo [9] the flux density increases at one end of the pole and decreases at the other, maintaining the average flux density. Should the flux density become too low at one end of the pole, permanent de-magnetization can occur. A frequent cause of de-magnetization is over-current at starting or during deceleration. A reduction of flux density leads to reduction of torque constant by (3.8).

Table 3.3 compares measurements of torque constant of the robot used in this work, with those obtained by other researchers for other Puma 560 robots. The values in the column headed 'CSIRO' were obtained by a colleague using the common technique of applying known loads to robot joints in position control mode and measuring the current required to resist that load. Values in the column headed 'Armstrong' were computed from the maximum torque and current data in [2]. Tarn [17] gives the torque constant for the base axis motors as 0.259 Nm/A (apparently from the manufacturer's specification). The Kawasaki data indicate torque constants of 0.253 and 0.095 Nm/A for base and wrist motors respectively.

Considerable variation is seen in Table 3.3, but all values for the base motor are less than Tarn's value. This may be due to loss of motor magnetization in the robots investigated, compared to the "as new" condition of the servo motors. Another source of error is the measurement procedure itself — since the joint is not moving the load torque is resisted by both the motor and the stiction mechanism. Lloyd [12] used an elaborate procedure based on the work involved in raising and lowering a mass so as to cancel out the effects of friction.

3.2.3 Armature impedance

Figure 3.4 shows the motor armature impedance $R_m + sL_m$, where R_m is the resistance due to the armature winding and brushes, and L_m is inductance due to the armature winding. For the Puma 560 armature inductance is low (around 1 mH [19]) and of little significance since the motor is driven by a current source.

Armature resistance, R_m, is significant in determining the maximum achievable joint velocity by (3.22) but is difficult to measure directly. Resistance measurement of a static motor exhibits

Axis	Expt.	Kawasaki	Tarn
Base	2.1	1.6	1.6
Wrist	6.7	3.83	-

Table 3.4: Comparison of measured and manufacturer's values of armature resistance (Ω). Experimental results obtained using (3.10).

a strong motor position dependence due to brush and commutation effects. A conventional *locked-rotor test* also suffers this effect and introduces the mechanical problem of locking the motor shaft without removing the motor from the robot. Measurements on a moving motor must allow for the effect of back EMF. Combining (3.6) and (3.3) and ignoring inductance we can write

$$\frac{v_m}{v_s} = \frac{R_m + R_s}{R_s} + \frac{K_m^2 \dot{\theta}}{R_s \tau_m} \tag{3.10}$$

where the second term represents the effect of back EMF, which may be minimized by increasing the torque load on the motor, and reducing the rotational speed. v_m and v_s are directly measurable and were fed to an FFT analyzer which computed the transfer function. The system exhibited good coherence, and the low frequency gain was used to estimate R_m since R_s is known. The resistance values for the base and wrist motors determined in this fashion are summarized in Table 3.4 along with manufacturer's data for the 'similar' Kawasaki Puma motors. The experiments give the complete motor circuit resistance including contributions due to the long umbilical cable, internal robot wiring, and connectors.

3.3 Current loop

The Unimate current loop is implemented in analog electronics on the so called *analog servo* board, one per axis, within the controller backplane. A block diagram of the Unimate current loop is shown in Figure 3.5. The transfer function of the block marked *compensator* was determined from analysis of schematics [20] and assuming ideal op-amp behaviour. The compensator includes an integrator, adding an open-loop pole at the origin in order to achieve Type 1 system characteristics in current following. The remaining dynamics are at frequencies well beyond the range of interest, and will be ignored in subsequent modeling. The voltage gain of the block marked *power amplifier*, $-k$, has been measured as approximately -50.

Current feedback is from a shunt resistor in series with the motor. The shunts are $0.2\,\Omega$ and $0.39\,\Omega$ for the base and wrist joints respectively. The high forward path gain, dominated by the compensator stage, results in the closed-loop gain, K_i, being governed by the feedback path

$$K_i = \frac{I_m}{V_{I_d}} = \frac{-1}{6.06 \times R_s} \tag{3.11}$$

The measured frequency response of the joint 6 current loop is shown in Figure 3.6, and the magnitude and phase confirm the analytic model above. The response is flat to 400 Hz which is consistent with Armstrong's [4] observation that current steps to the motor settle in less than $500\,\mu$s. The use of a current source to drive the motor effectively eliminates the motor's electrical pole from the closed-loop transfer function.

The measured current loop transconductance of each axis is summarized in Table 3.5. The gains are consistently higher than predicted by (3.11), but no more than expected given the tolerance of components used in the various gain stages[3]. The maximum current in Table 3.5 is estimated from the measured transconductance and the maximum current loop drive of $v_{i_d} = 10\,\mathrm{V}$.

[3] The circuitry uses only standard precision, 10%, resistors.

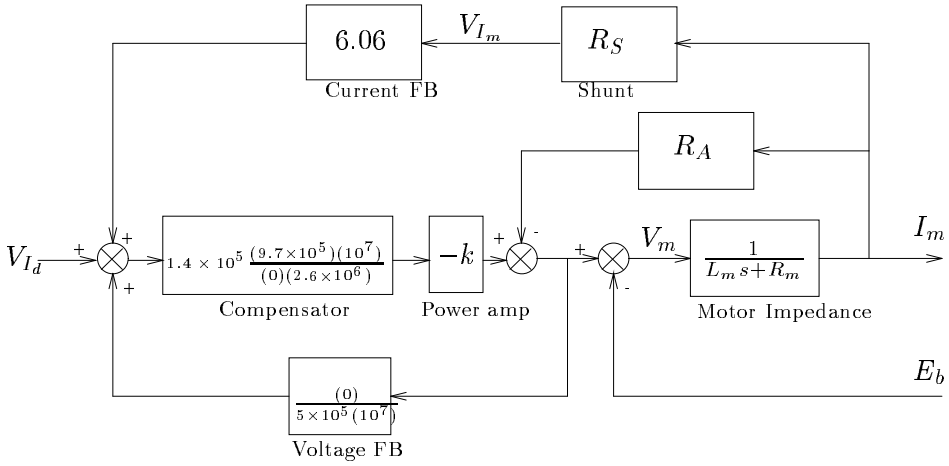

Figure 3.5: Block diagram of motor current loop. E_b is the motor back EMF, V_m motor terminal voltage, and R_A the amplifier's output impedance.

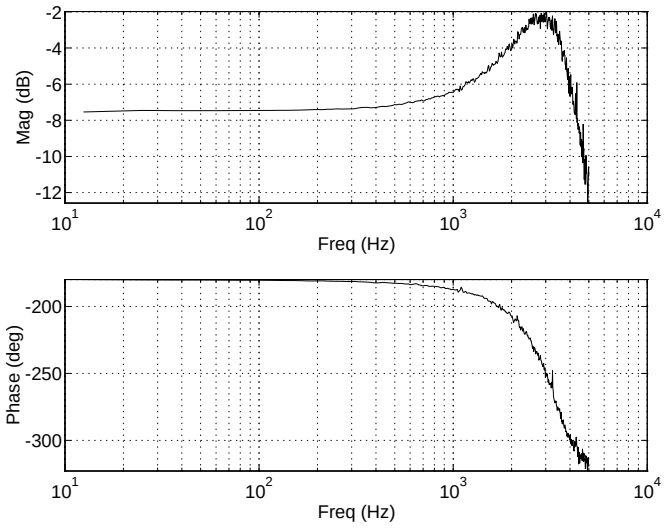

Figure 3.6: Measured joint 6 current loop frequency response I_m/V_{I_d}.

3.4 Combined motor and current-loop dynamics

The measured transfer function between motor current and motor velocity for joint 6 is given in Figure 3.7, along with a fitted transfer function. The transfer function corresponds to a linearization of the non-linear model of Figure 3.3 for a particular level of excitation. The fitted model is

$$\frac{\Omega_m}{V_{I_d}} = \frac{24.4}{(31.4)} \text{ rad/s/V} \tag{3.12}$$

which has a pole at 5Hz. From Figure 3.3 and (3.11) the model response is given by

$$\frac{\Omega_m}{V_{I_d}} = \frac{K_m K_i}{J_{eff}s + B_{eff}} \tag{3.13}$$

where J_{eff} and B_{eff} are the effective inertia and viscous friction due to motor and transmission. From the measured break frequency and Armstrong's inertia value [2] of $33 \times 10^{-6}\,\text{kgm}^2$ the

Joint	K_i (A/V)	$i_{m_{max}}$ (A)
1	-0.883	8.83
2	-0.877	8.77
3	-0.874	8.74
4	-0.442	4.42
5	-0.442	4.42
6	-0.449	4.49

Table 3.5: Measured current loop transconductances and estimated maximum current. The measurement is a lumped gain from the DAC terminal voltage to current loop output.

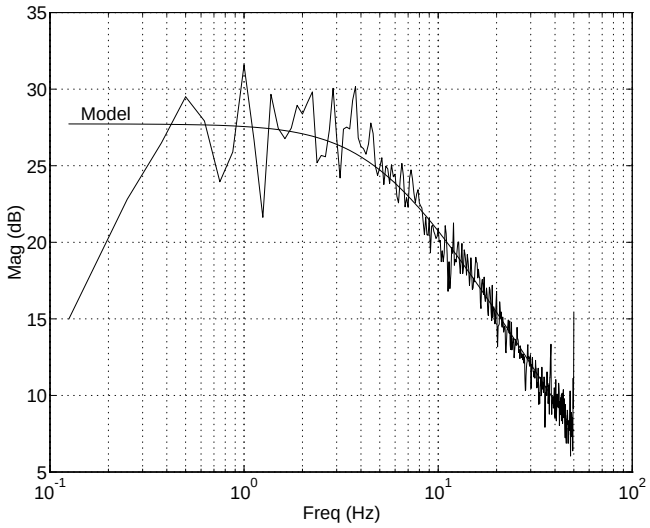

Figure 3.7: Measured joint 6 motor and current loop transfer function, Ω_m/V_{I_d}, with fitted model response.

estimated effective viscous friction is

$$B_{eff} = 33 \times 10^{-6} \times 31.4 = 1.04 \times 10^{-3}\,\text{Nms/radm} \tag{3.14}$$

Substituting these friction and inertia values into equation (3.13) gives

$$\frac{\Omega_m}{V_{I_d}} = \frac{K_m K_i}{(33 \times 10^{-6})s + (1.04 \times 10^{-3})} \tag{3.15}$$

Equating the numerator with (3.12), and using known K_i from Table 3.5 leads to an estimate of torque constant of 0.060.

The estimated effective viscous friction is significantly greater than the measured value of 37×10^{-6} Nms/rad from Table 3.1. This is a consequence of the linear identification technique used which yields the 'small-signal' response. The effective viscous friction includes a substantial contribution due to Coulomb friction particularly at low speed. Coulomb friction may be linearized using sinusoidal or random input describing functions giving an effective viscous damping of

$$B_{eff} = B + \frac{k\tau_C}{\sqrt{2}\sigma_{\dot{\theta}}} \tag{3.16}$$

where $\sigma_{\dot{\theta}}$ is the RMS velocity, and $k = 1.27$ (sinusoidal) or $k = 1.13$ (random).

Joint	τ_{Iloop}	τ_{fuse}
1	120	56
2	200	97
3	110	52
4	22	10
5	22	10
6	21	10

Table 3.6: Maximum torque (load referenced) at current loop and fuse current limits. All torques in Nm.

The large-signal response may be measured by step response tests, and these indicate a much higher gain — approximately 200 rad/s/V. From (3.13) the DC gain is

$$\frac{K_m K_i}{B_{eff}} \tag{3.17}$$

which leads to an estimate of effective viscous friction of 126×10^{-6} Nms/rad. As expected at the higher speed, this estimate is closer to the measured viscous friction coefficient of Table 3.1.

3.4.1 Current and torque limits

Maximum motor current provides an upper bound on motor torque. When the motor is stationary, the current is limited only by the armature resistance

$$i_{max} = \frac{v_a}{R_m} \tag{3.18}$$

where v_a is the maximum amplifier voltage which is 40V for the Puma. From the armature resistance data given in Table 3.4 it can be shown that the maximum current given by (3.18) is substantially greater than the limit imposed by the current loop itself. Maximum current is thus a function of the current loop, not armature resistance. The sustained current is limited further by the fuses, or circuit breakers, which are rated at 4A and 2A for the base and wrist motors respectively. The fuse limits are around half the maximum achievable by the current loop.

Using maximum current data from Table 3.5, and known motor torque constants, the maximum joint torques can be computed. Table 3.6 shows these maxima for the case of current limits due to the current loop or fuse.

3.4.2 Back EMF and amplifier voltage saturation

A robot manipulator with geared transmission necessarily has high motor speeds, and thus high back EMF. This results in significant dynamic effects due to reduced motor torque and amplifier voltage saturation. Such effects are particularly significant with the Puma robot which has a relatively low maximum amplifier voltage. Figure 3.8 shows these effects very clearly — maximum current demand is applied but the actual motor current falls off rapidly as motor speed rises. Combining (3.9) and (3.6), and ignoring motor inductance since steady state velocity and current are assumed, the voltage constraint can be written as

$$|\dot{\theta} K_m + \frac{\tau}{K_m} R_T| \leq v_a \tag{3.19}$$

where R_T is the total circuit resistance comprising armature resistance, R_m and amplifier output impedance R_a. Rising back EMF also diminishes the torque available from the actuator during

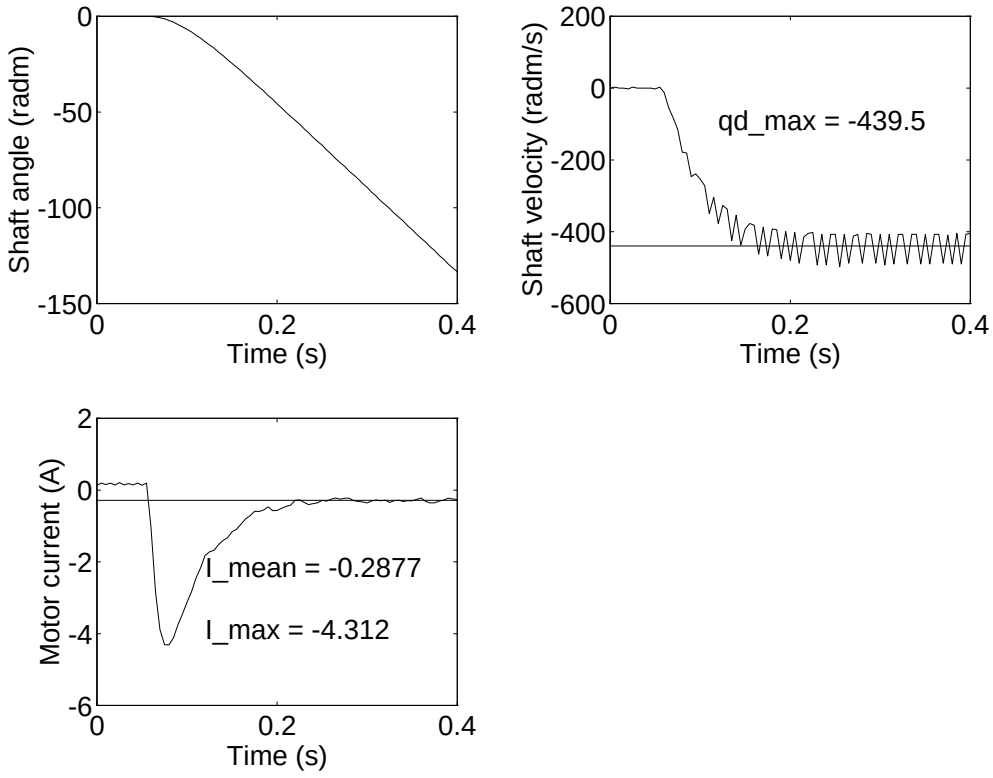

Figure 3.8: Measured motor and current loop response for a step current demand of $V_{I_d} = 10$. Motor angle, velocity and current versus time are shown. $\dot{q}_{max}$ is the steady state velocity, and I_{mean} is the steady state current. The finite rise-time on the current step response is due to the anti-aliasing filter used. The sampling interval is 5ms.

acceleration

$$\tau_{avail} = (v_a - \dot{\theta} K_m) \frac{K_m}{R_T} \tag{3.20}$$

However during deceleration, back EMF works to increase motor current which is then limited by the current loop or fuse. When the frictional torque equals the available torque

$$\tau_C + \dot{\theta} B = (v_a - \dot{\theta} K_m) \frac{K_m}{R_T} \tag{3.21}$$

the joint velocity limit due to amplifier voltage saturation

$$\dot{\theta}_{vsat} = \frac{v_a K_m - R_T \tau_C}{R_T B + K_m^2} \tag{3.22}$$

is attained. Armature resistance, R_T, and friction, serve to lower this value below that due to back EMF alone. The corresponding motor current is

$$i_{vsat} = \frac{B v_a + K_m \tau_C}{R_T B + K_m^2} \tag{3.23}$$

Experiments were conducted in which the maximum current demand was applied to each current loop, and the motor position and current history recorded, as shown in Figure 3.8. The initial current peak is approximately the maximum current expected from Table 3.5, but falls

Joint	Measured		Estimated	
	$\dot{\theta}_{vsat}$	i_{vsat}	$\dot{\theta}_{vsat}$	i_{vsat}
1	120	2.1	149	3.2
2	163	0.26	165	1.3
3	129	0.42	152	1.7
4	406	0.56	534	0.84
5	366	0.39	516	0.75
6	440	0.29	577	0.51

Table 3.7: Comparison of experimental and estimated velocity limits due to back EMF and amplifier voltage saturation. Velocity and current limits are estimated by (3.22) and (3.23) respectively. Velocity in radm/s and current in Amps.

off due to voltage saturation as motor speed rises. Without this effect the motor speed would ultimately be limited by friction. Table 3.7 compares computed maximum joint velocities and currents with the results of experiments similar to that leading to Figure 3.8. The estimates for $\dot{\theta}_{vsat}$ are generally higher than the measured values, perhaps due to neglecting the amplifier output resistance (which has not been measured). Measurements for joints 2 and 3 are complicated by the effect of gravity which effectively adds a configuration dependent torque term to (3.20). To counter this, gravity torque was computed and the corresponding current subtracted from that measured. At voltage saturation, the current is generally around 30% of the maximum current achievable by the power amplifier.

3.4.3 MATLAB simulation

A complete SIMULINK model of the motor and current loop is shown in Figure 3.9. This model is a stiff non-linear system and is thus very slow to simulate. The high-order poles due to motor inductance and the current loop compensator are around 300 times faster than the dominant mechanical pole of the motor. Non-linearities are introduced by voltage saturation and Coulomb friction. The reduced order model, Figure 3.10, has similar non-linear characteristics but does not have the high-order poles, is much faster to integrate, and is to be preferred for simulation purposes. Increasing the value of R_m above that in Table 3.4 allows the model to more accurately predict the saturation speed and current. This could be considered as allowing for the currently unmodeled amplifier output impedance.

3.5 Velocity loop

Like the current loop, the Unimate velocity loop is implemented in analog electronics on the *analog servo* board. A block diagram of the velocity loop is shown in Figure 3.11, and includes the motor and current loop blocks already discussed. The gains K_{gain} and K_{vel} are set by trimpots R31 (GAIN) and R29 (VELOCITY) respectively on each analog servo board. The Unimation Puma does not use a tachometer to measure motor velocity. Instead a velocity signal is synthesized from the motor's incremental encoder pulses using a simple frequency to voltage converter, since the pulse frequency is proportional to shaft velocity. The gain of the digital tachometer has been experimentally determined to be

$$K_{tach} = \frac{V_{\Omega_m}}{\Omega_m} = 34 \times 10^{-3} \,\mathrm{Vs/radm} \tag{3.24}$$

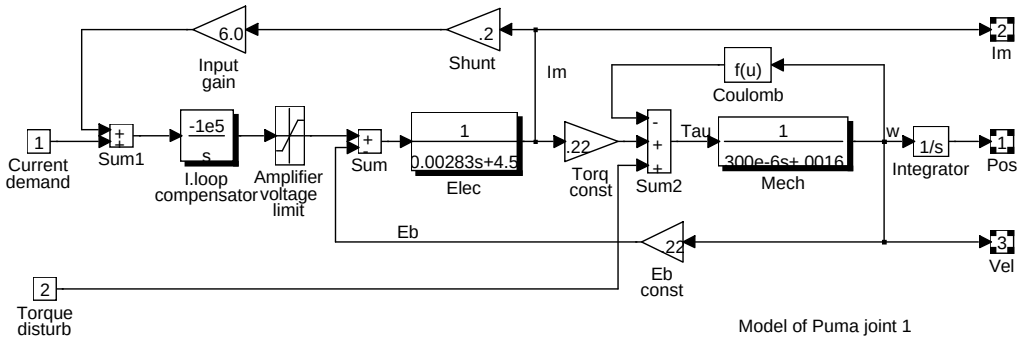

Figure 3.9: SIMULINK model MOTOR: joint 1 motor and current loop dynamics.

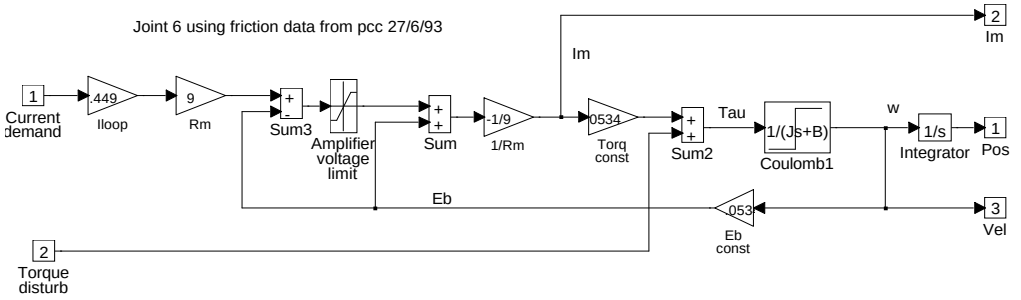

Figure 3.10: SIMULINK model LMOTOR: reduced order model of joint 6 motor and current loop dynamics.

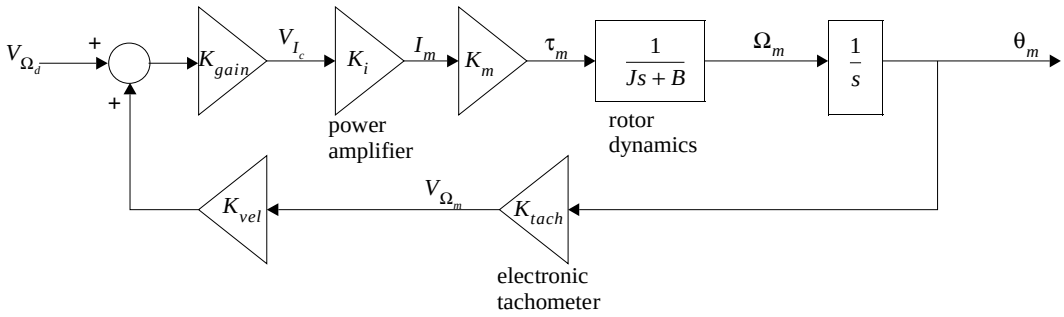

Figure 3.11: Velocity loop block diagram.

with some increase in gain at frequencies below 2Hz. From Figure 3.11 the closed loop transfer function for the motor, current and velocity loop is

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{K_{gain}K_iK_m}{Js + B + \underbrace{K_iK_mK_{gain}K_{vel}K_{tach}}_{B'}} \tag{3.25}$$

It is desirable that $B' \gg B$ in order that the closed-loop dynamics are insensitive to plant parameter variation, but in practice this is not the case. The adjustable gains K_{gain} and K_{vel} provide interacting control of the DC gain and closed-loop pole location. The measured transfer function between motor velocity and velocity demand for joint 6 is given in Figure 3.12 and the fitted model is

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{-11.9}{(148.6)}\,\text{radm/s/V} \tag{3.26}$$

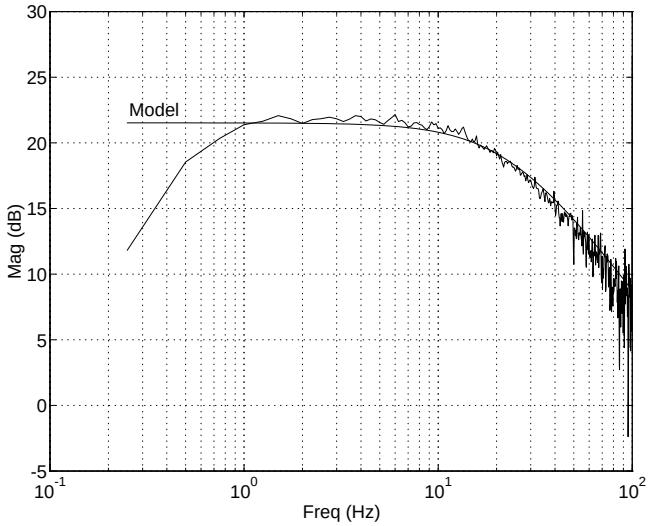

Figure 3.12: Measured joint 6 velocity loop transfer function, Ω_m/V_{Ω_m}, with fitted model response. Phase data is not shown but is indicative of a sign reversal.

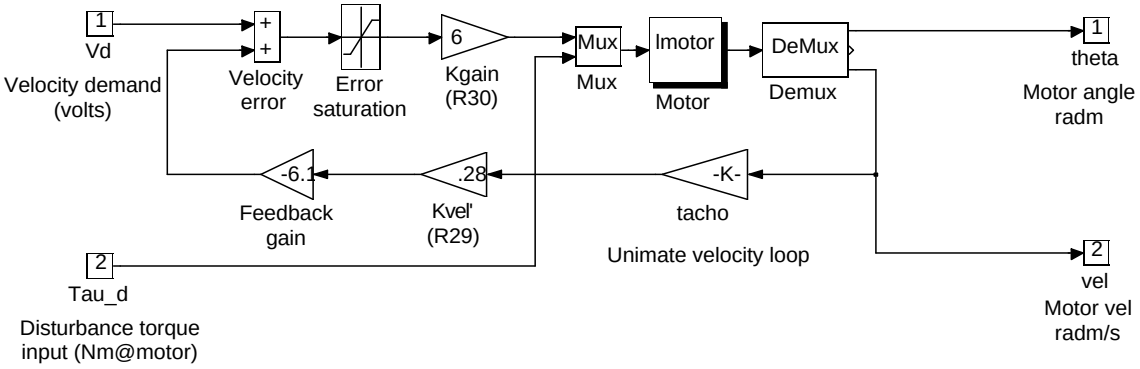

Figure 3.13: SIMULINK model VLOOP: motor velocity loop.

Observe that the mechanical pole of the motor at 5Hz has been 'pushed out' to nearly 25Hz by the action of the velocity loop. Substituting known numeric values[4] into (3.25) results in the transfer function

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{-12.8}{(137)} \text{ radm/s/V} \tag{3.27}$$

and compares well with the measured result (3.26). Again, this transfer function represents the small-signal response of the system.

The large signal gain for each axis has also been determined experimentally by providing a step demand to the velocity loop and measuring the resultant slope of the joint position versus time curve. These results, summarized in Table 3.8, are highly dependent on the way the particular analog velocity loop has been 'tuned', as given by (3.25). It can be seen that joint 2 has a significantly higher velocity gain than the others, probably to compensate for its much higher gear reduction ratio.

A SIMULINK model of the velocity loop is shown in Figure 3.13. This model makes use of the motor and current loop model LMOTOR developed earlier.

[4]The gains $K_{gain} = 2.26$, and $K_{vel} = 1.75$ were determined by measuring the transfer function between adjacent points in the circuitry with an FFT analyzer. $K_{vel} = 6.06K'_{vel}$ where $K'_{vel} = 0.288$ is the gain of the trimpot R29.

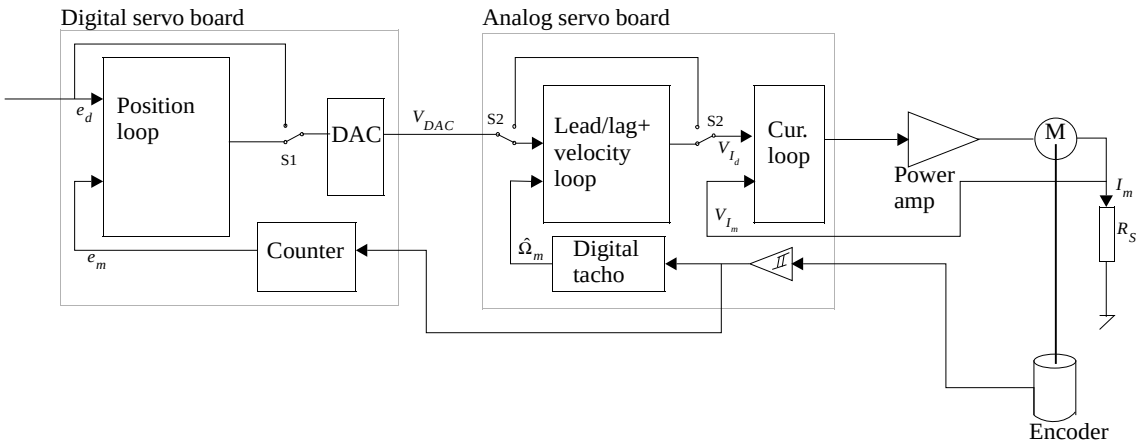

Figure 3.14: Unimation servo position control mode. The switch S2 routes the output of the DAC to the lead compensator and velocity loop.

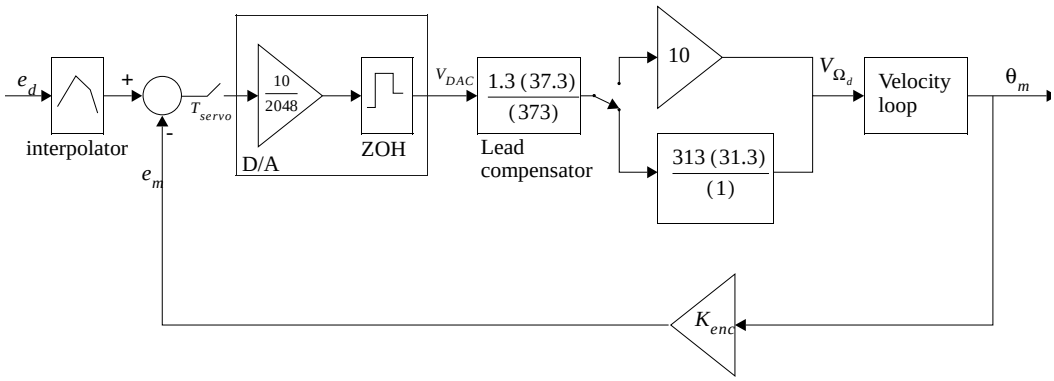

Figure 3.15: Block diagram of Unimation position control loop.

3.6 Position loop

The position loop is implemented on a *digital servo* board — one per axis. This board utilizes a 6503, 8 bit microprocessor clocked at 1MHz to close an axis position loop at approximately 1kHz, and also to execute commands from the host computer. Position feedback is from incremental encoders, whose characteristics are summarized in Table A.3, fitted to the motor shafts. The analog control signal, v_{DAC}, is generated by a 12-bit DAC with a bipolar output voltage in the range -10V to 9.995V and drives, via an analog compensation network, the analog velocity demand. The overall control system structure is shown in Figure 3.14.

A block diagram of the position loop is given in Figure 3.15. The digital control loop operates at a sample interval of $T_{servo} = 924\,\mu$s. The encoder demand, e_d, is provided by the host computer at intervals of $2^N\,T_{servo}$ where N is in the range 2 to 7 resulting in host setpoint intervals of approximately 4.5, 7, 14, 28, 56 or 112 ms. The microprocessor linearly interpolates between successive position setpoints, at the servo rate. At that same rate, the microprocessor implements the fixed-gain control law[5]

$$v_{DAC} = \frac{10}{2048}(e'_d - e_m) \tag{3.28}$$

where e'_d is the desired encoder value from the interpolator, and e_m is the measured encoder count. This control law has been verified by disassembly of the microprocessor's EPROM and

[5] Since the digital controller has a fixed gain, loop gain is adjusted by the velocity loop gain control.

Joint	ω/v_{DAC} radm/s/V	$\dot{\theta}_{max}$ radm/s	$\dot{\theta}_{max}/\dot{\theta}_{vsat}$ radm/s
1	-101	89	74%
2	-318	281	172%
3	-90.2	80	62%
4	-366	324	80%
5	-235	208	57%
6	-209	185	42%

Table 3.8: Measured step-response gains, ω/v_{DAC} of velocity loop. Step magnitude was selected so as to avoid voltage saturation effects. These gain values include the effect of the position-loop lead network and switchable integral/gain stage. $\dot{\theta}_{max}$ is the estimated maximum velocity due to the velocity loop when $v_{\Omega_d} = 10\,\mathrm{V}$. Rightmost column is ratio of velocity limits due to velocity loop and voltage saturation effects.

by measuring the transfer function V_{DAC}/e_m. The output voltage is a velocity command which is conditioned by a phase lead network

$$1.3\frac{(37.3)}{(373)} \tag{3.29}$$

so as to increase the closed-loop bandwidth. The gain of 1.3 was determined by analysis of the circuit, but the measured gain for joint 6 was found to be 1.13. A switchable integral action stage

$$I(s) = \begin{cases} 10 & \text{if the robot is moving, or} \\ 313\frac{(31.3)}{(1)} & \text{if the robot is 'on station'} \end{cases} \tag{3.30}$$

is enabled by the microprocessor when the axis is within a specified range of its setpoint. Integral action boosts the gain at low frequency, increasing the disturbance rejection.

The transfer function of the lead network (3.29) and integral stage (3.30) introduce a gain of around 11.3 before the velocity loop. The velocity command voltage, V_{Ω_d}, is limited by the circuit to the range $\pm 10\,\mathrm{V}$, so the DAC voltage must in turn be limited such that

$$|v_{DAC}| \leq \frac{10}{11.3} \tag{3.31}$$

This limits the usable voltage range from the DAC to only $\pm 0.88\,\mathrm{V}$ — that is only 9% of the available voltage range, and gives an effective resolution of less than 8 bits on velocity demand. This saturation of the velocity loop demand voltage is readily apparent in experimentation. From measured velocity loop gains, ω/v_{DAC}, shown in Table 3.8 and knowledge of maximum DAC voltage from (3.31), the maximum joint velocities can be determined. These maxima are summarized in Table 3.8 along with the ratio of velocity limits due to velocity loop and voltage saturation effects. For most axes the maximum demanded joint velocity is significantly less than the limit imposed by voltage saturation. The only exception is joint 2 which, as observed earlier, has an abnormally high velocity-loop gain.

Root-locus diagrams for the joint 6 position loop are shown in Figures 3.16 and 3.17. For the case with no integral action the dominant pole is on the real axis due to the open-loop pole at the origin moving toward the compensator zero, resulting in a closed-loop bandwidth of 32 Hz. The complex pole pair has a natural frequency of 57 Hz and a damping factor of 0.62. With integral action enabled the dominant mode is a lightly damped complex pole pair with a natural frequency of around 1.2 Hz and a damping factor of 0.25.

A SIMULINK model of the position controller is shown in Figure 3.18. The structure of the switchable integral action stage is somewhat different to Figure 3.15 but more closely represents

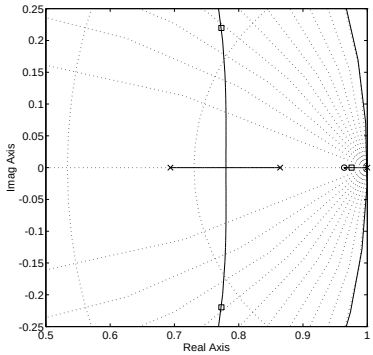

Figure 3.16: Root-locus diagram of position loop with no integral action.

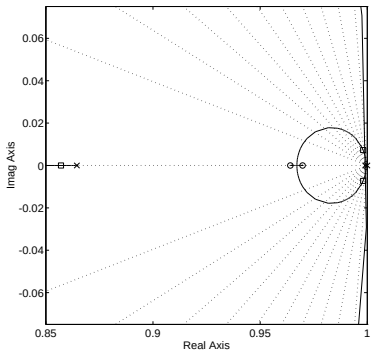

Figure 3.17: Root-locus diagram of position loop with integral action enabled.

the actual controller, in particular with respect to 'bumpless' switching of the integrator. This model is used extensively in later chapters when investigating the behaviour of visual-loop closed systems.

3.6.1 Host current control mode

The Unimate digital servo board also allows the motor current to be controlled directly by the host. In this mode the DAC output voltage is connected directly to the current loop as shown in Figure 3.19. The position loop is inactive after a CURMODE command, and the velocity loop is disabled by deasserting /SERVO_ENABLE. The command voltage from the D/A converter is fed straight to the current loop summing junction via an analog switch enabled by the /CURRENT_MODE signal. The gain of the D/A is 10/2048, and the gain of the current loop is given in Table 3.5. The DAC is updated within T_{servo} of the setpoint being given. This mode is useful when the host computer implements its own axis control strategy.

3.6.2 LSB servo mode

A curious feature of the servo system is the so-called LSB servo mode, controlled by the /LSB_SERVO_MODE output of the digital servo board. This feature is not used by the existing digital servo firmware.

A voltage, strongly proportional to the displacement from the current encoder edge is synthesized from the analog quadrature encoder signals by U5, U6 and U8. This exerts a restoring force to move the shaft back to the encoder edge, achieving levels of accuracy not possible digitally due to quantization effects.

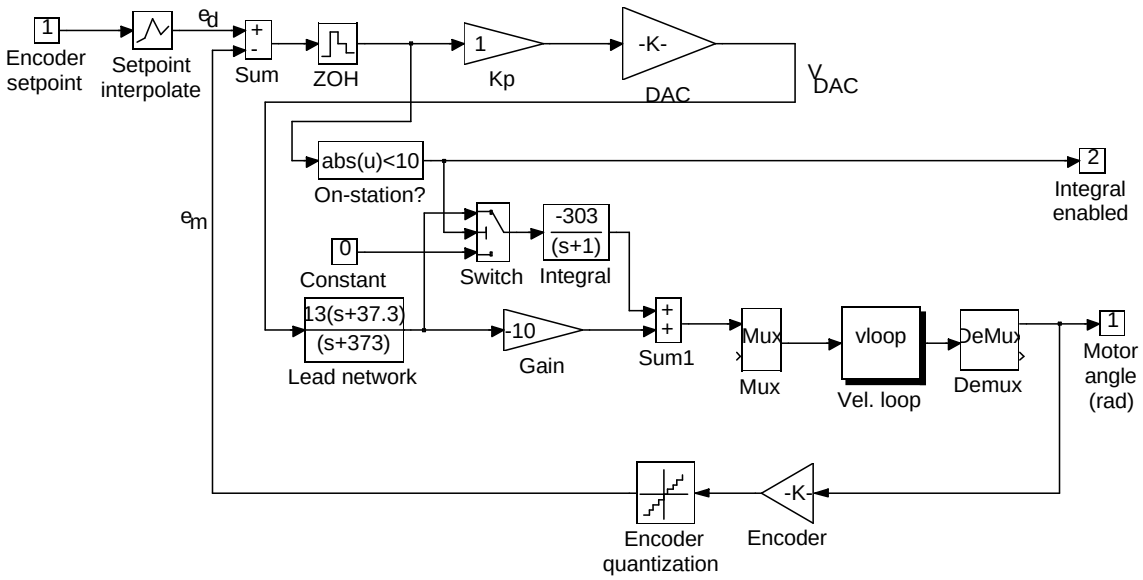

Figure 3.18: SIMULINK model POSLOOP: Unimation digital position control loop.

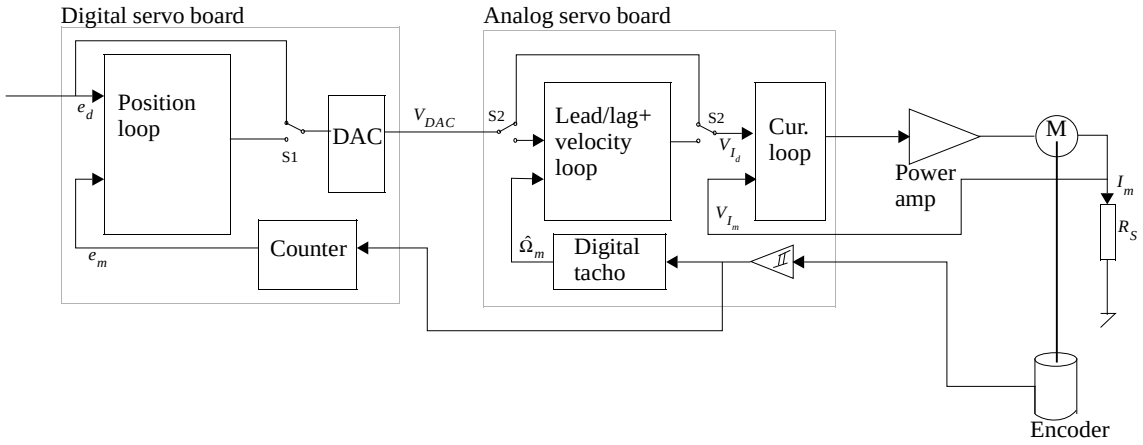

Figure 3.19: Block diagram of Unimation servo current control mode. The switch S2 is used to route the host setpoint directly to the current loop demand via the DAC, bypassing the velocity loop and lead compensator.

3.6.3 Servo jitter

In position mode the motor shaft is generally not stationary, but vibrating at relatively high frequency. This is likely to be due to the servo loop interacting with non-linear friction forces on the motor shaft, or quantization effects in the digital controller. This low level, high frequency vibration can cause significant contributions to end-effector sensed force and acceleration.

Appendix A

The Puma 560 manipulator

A.1 Kinematic parameters

The Puma 560 kinematics are dependent upon four non-zero length parameters and a number of axis twist angles which are taken as either exactly 0° or exactly 90°.

These lengths, which may be measured directly, are:

1. Distance between shoulder and elbow axes along the upper arm link
2. Distance from the elbow axis to the center of spherical wrist joint along the lower arm
3. Offset between axis of joint 4 and the elbow
4. Offset between the waist axis and the joint 4 axis.

The kinematic parameters for the Puma 560 are given in Table A.1. Since there are many ways of assigning coordinate frames to the manipulator, a set of kinematic parameters must be taken in the context of a particular set of axis and angle conventions; we use the conventions of Paul and Zhang [15]. The α values used are negative compared to others such as Lee [10]. This is due to the definition of a right-handed configuration for the zero joint angle pose used by Paul and Zhang. The transform T6 is considered as the position/orientation of the center of the wrist mechanism. The distance from wrist center to the flange plate is given by Lee [10] as 56.25mm.

Typically the robot pose for zero joint angles is a left-handed configuration with the upper arm horizontal along the X axis and the lower arm vertically upwards. However, Paul and Zhang adopt a similar pose but in a right-handed configuration. Thus by their convention, the upright (VAL READY) position is given by $\underline{\theta} = [0\ 90\ -90\ 0\ 0\ 0]^\circ$ whilst for others it is $\underline{\theta} = [0\ -90\ 90\ 0\ 0\ 0]^\circ$.

i	α_i	A_i	D_i
1	90	0	0
2	0	431.8	0
3	-90	19.1	125.4
4	90	0	431.8
5	-90	0	0
6	0	0	0

Table A.1: Kinematic constants for Puma 560. α in degrees, A and D are in mm.

Joint	Armstrong [2]	BreakingAway [21]
G_1	62.61	-62.6111
G_2	107.36	107.815
G_3	53.69	-53.7063
G_4	76.01	76.03636
G_5	71.91	71.923
G_6	76.63	76.686
G_{45}	$-1/G_5$	
G_{46}	$-1/G_6$	
G_{56}	$-13/72$	

Table A.2: Puma 560 gear ratios

A.2 Gear ratios

The sign of the ratio indicates the direction of joint angle, θ, change given a positive increase in digital position loop encoder count. In servo current control mode a positive current causes motion in the negative encoder count direction. It seems likely that Armstrong et al. [2] have given only the absolute value of gear ratio. Nagy's [14] gear ratios for the Kawasaki Puma 560, agree with [21].

The cross-coupling values have been determined from examination of the wrist mechanism, and correspond with numerical values given by [21]. The relationship between joint and motor angles is

$$\underline{\theta_j} = \begin{bmatrix} \frac{1}{G_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{G_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{G_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_4} & 0 & 0 \\ 0 & 0 & 0 & \frac{G_{45}}{G_4} & \frac{1}{G_5} & 0 \\ 0 & 0 & 0 & \frac{G_{46}}{G_4} & \frac{G_{56}}{G_5} & \frac{1}{G_6} \end{bmatrix} \underline{\theta_m}$$

A.3 Encoders

Joint	Encoder resolution	Counts/motor rev.	Counts/radl
1	250	1000	9965
2	200	800	13727
3	250	1000	8548
4	250	1000	12102
5	250	1000	11447
6	250	500†	6102

Table A.3: Puma 560 joint encoder resolution. †Note that joint 6 is generally run in 'divide by two mode', so that the encoder count maintained by the digital servo board is half the number of actual encoder counts. This is necessary since joint 6 can rotate through a range of more than 2^{16} counts.

Appendix B

Encoder calibration

At reset all joint encoders are set to 0x8000, the middle of the 16 bit unsigned integer range. By convention this encoder value corresponds to the robot pose where the robot arm is vertical (VAL ready position).

The incremental encoders respond only to change in position – the absolute encoder values corresponding to the pose of the robot at startup are not known and are determined by the calibration procedure.

Each joint also has a potentiometer sensor, which is a low resolution absolute shaft angle encoder. The calibration procedure involves

1. moving each joint to a zero index mark (one per motor revolution) using the 'stop at zero index' control bits in the joint status word.

2. reading the potentiometer sensor, ρ[1], and determining the approximate encoder value from the known potentiometer response;

$$\hat{e}_i = \alpha_i \rho_i + \beta_i$$

   where α_i and β_i, are experimentally determined potentiometer characteristics.

3. We know that the encoder value is of the form

$$EFIRST + n \times EDELTA$$

   where EFIRST is the encoder count at the first zero index away from the 'READY' position, EDELTA is the number of encoder counts between zero indices, and n is an integer. Thus given an approximate encoder count, $\hat{e}$, derived from the potentiometer above, the exact count is given by

$$e = nint(\frac{\hat{e} - EFIRST}{EDELTA}) EDELTA + EFIRST$$

   where $nint(x)$ gives the integer nearest to x.

4. Finally, the encoder registers in the digital servo boards are set to their correct value with the SETPOS command.

The coefficients for each potentiometer, α_i and β_i, are determined experimentally. The procedure involves

1. calibrating the robot as accurately as possible, generally via the engraved markings on each joint, or using plumb bobs, levels and theodolites

[1] To reduce the effects of quantization error, ρ is often the sum of N repeated readings

2. measuring ρ_i and e_i at every zero index over the working range of the joint

3. performing a regression on the ρ_i, e_i pairs.

The potentiometer calibration procedure need only be done once, and repeated only after modification or repair to any of the joint motor/potentiometer assemblies.

Appendix C

Communications protocol

The following 'C' code illustrate the lowest level primitives required for communications with the Unimate servo subsystem. The argument `cmd` contains the command code, and for non-vector operations also the joint address.

In this example the interface is memory mapped, with one word for data read/write, and one word for reading/writing status lines to the AIB.

```
#define TIMEOUT         50
#define NLOOPS          128

#define VECBIT          0x80

#define DATA            pp->status|=STATCSR0
#define COMMAND         pp->status&=~STATCSR0

#define OUTW(w)         pp->data = w
#define INW(w)          pp->data

struct port {
        unsigned short  data;
        unsigned short  status;
};

/* define status word bits */

#define STATREQA        0x01
#define STATREQB        0x02
#define STATCSR0        0x04
#define STATCSR1        0x08

struct port             *pp = HOSTPORTADDR;

/*
 * return the result of 'cmd' applied to a single joint
 */
ReadSingle(cmd)
int     cmd;
{
        int     r;
```

```
        COMMAND;
        OUTW(cmd);
        DATA;
        if ((cmd&7) != SELECT7)
                ready("rsingle2", cmd);
        r = INW;
        if ((cmd&7) != SELECT7)
                ready("rsingle3", cmd);
        return r&0xffff;
}

/*
 * execute ‘cmd’ on a single joint
 */
WriteSingle(cmd, data)
int             cmd;
unsigned int    data;
{
        COMMAND;
        OUTW(cmd);
        DATA;
        OUTW(data);
        if ((cmd&7) != SELECT7)
                ready("wsingle", cmd);
}

/*
 * return the result of ‘cmd’ applied to all joints
 */
ReadVector(cmd, vec)
int             cmd;
unsigned int    *vec;
{
        int     i, j;

        COMMAND;
        OUTW(cmd|VECBIT);
        DATA;
        for (i=0; i<NJOINTS; i++) {
                ready("rvector", cmd);
                vec[i] = INW;
        }
}

/*
 * execute ‘cmd’ on all joints, different data for each joint
 */
WriteVector(cmd, vec)
int     cmd;
```

```
unsigned int    *vec;
{
        int     i;

        COMMAND;
        OUTW(cmd | VECBIT);
        DATA;
        for (i=0; i<NJOINTS; i++) {
                OUTW(cmd==STOPMDE ? 0 : vec[i]);
                ready("wvector", cmd);
        }
}

/*
 * execute ‘cmd’ on all joints, same data for each joint
 */
WriteMultiple(cmd, val)
int             cmd;
unsigned int    val;
{
        int     i;

        COMMAND;
        OUTW(cmd | SEQBIT);
        DATA;
        for (i=0; i<NJOINTS; i++) {
                OUTW(cmd==STOPMDE ? 0 : val);
                ready("wmultiple", cmd);
        }
}

/*
 * return attention and joint tolerance bits
 */
GetTol()
{
        COMMAND;
        OUTW(0);
        return INW;
}

/*
 * wait until servos acknowledge command completion
 */
static  ready(where, cmd)
char    *where;
int     cmd;
{
        int     timeout;
```

```
        timeout = TIMEOUT;
        while ((pp->status & STATREQA) == 0 && --timeout > 0)
                ;
        if (timeout == 0) {
                printf("joint timeout from %s, cmd = %x\n", where, cmd);
                return;
        }
}
```

C.1 Initialization

Before the digital servos can be used some initialization is required by the host.

1. Enable integration and servoing using the STDATA command to write the appropriate joint status word, see Table 2.5.
2. Put the servo into stop mode using the STOPMDE command.
3. Set the number of lines between zero indices using the STDATA command.
4. Set the velocity gain using the STDATA command.
5. Set the integration band tolerance using the INTTOL command.
6. Set the in-tolerance band using the POSTOL command.
7. Set the host setpoint update interval using the STDATA command and the data from Table 2.6.

Appendix D

Backplane connector notation

The pins on the backplane connectors are labeled using DEC conventions, as shown in Figure D.1. Pins within each connectors are labeled with letters from DECs connector alphabet, see Table D.1.

A	1	18
B	2	17
C	3	16
D	4	15
E	5	14
F	6	13
H	7	12
J	8	11
K	9	10
L	10	9
M	11	8
N	12	7
P	13	6
R	14	5
S	15	4
T	16	3
U	17	2
V	18	1

Table D.1: Letters/numbering convention used for backplane connectors

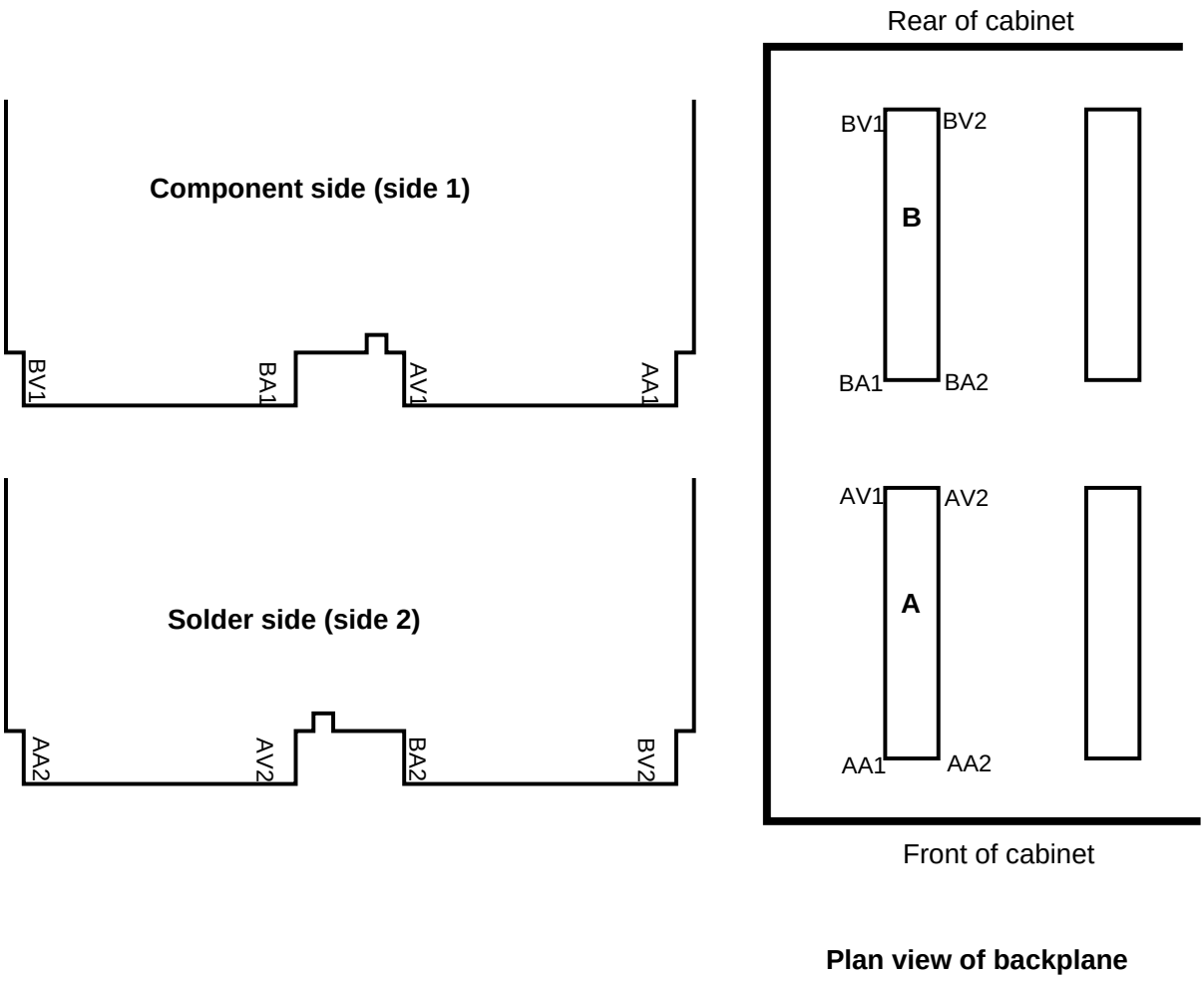

Figure D.1: Board connector labeling details

Bibliography

[1] G. B. Andeen, editor. *Robot Design Handbook*. McGraw-Hill, 1988.

[2] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the Puma 560 arm. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 1, pages 510–18, Washington, USA, 1986.

[3] Brian Armstrong. Friction: Experimental determination, modeling and compensation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1422–1427, 1988.

[4] B.S. Armstrong. *Dynamics for Robot Control: Friction Modelling and Ensuring Excitation During Parameter Identification*. PhD thesis, Stanford University, 1988.

[5] Peter I. Corke. *High-Performance Visual Closed-Loop Robot Control*. PhD thesis, University of Melbourne, Dept.Mechanical and Manufacturing Engineering, July 1994.

[6] P.I. Corke and B. Armstrong-Hélouvry. A note on the literature of PUMA 560 dynamics. *Robotica*, to appear.

[7] P.I. Corke and B.S. Armstrong-Hélouvry. A search for consensus among model parameters reported for the PUMA 560 robot. In *Proc. IEEE Int. Conf. Robotics and Automation*, San Diego, May 1994.

[8] Robert M. Goor. A new approach to minimum time robot control. Technical Report GMR-4869, General Motors Research Laboratories, November 1984.

[9] T. Kenjo and S. Nagamori. *Permanent-Magnet and Brushless DC motors*. Clarendon Press, Oxford, 1985.

[10] C. S. G. Lee. Robot arm kinematics, dynamics and control. *IEEE Computer*, 15(12):62–80, December 1982.

[11] M. Liu. Puma 560 robot arm analogue servo system parameter identification. Technical Report ASR-91-1, Dept. Mechanical and Manufacturing Engineering, University of Melbourne, February 1991.

[12] John Lloyd. Implementation of a robot control development environment. Master's thesis, Mc Gill University, December 1985.

[13] A. Melidy and A. A. Goldenberg. Operation of PUMA 560 without VAL. *Proc. Robots 9*, pages 18.61–18.79, June 1985.

[14] P. V. Nagy. The PUMA 560 industrial robot: Inside-out. In *Robots 12*, pages 4.67–4.79, Detroit, June 1988. SME.

[15] R. P. Paul and Hong Zhang. Computationally efficient kinematics for manipulators with spherical wrists. *Int. J. Robot. Res.*, 5(2):32–44, 1986.

[16] R.P. Paul, M. Rong, and H. Zhang. Dynamics of Puma manipulator. In *American Control Conference*, pages 491–96, June 1983.

[17] T. Tarn, A. K. Bejczy, X. Yun, and Z. Li. Effect of motor dynamics on nonlinear feedback robot arm control. *IEEE Trans. Robot. Autom.*, 7(1):114–122, February 1991.

[18] T. J. Tarn, A. K. Bejczy, S. Han, and X. Yun. Inertia parameters of Puma 560 robot arm. Technical Report SSM-RL-85-01, Washington University, St. Louis, MO., September 1985.

[19] T.J. Tarn, A.K. Bejczy, G.T. Marth, and A.K. Ramadorai. Performance comparison of four manipulator servo schemes. *IEEE Control Systems Magazine*, 13(1):22–29, February 1993.

[20] Unimation Inc., Danbury, CT. *Unimate PUMA 500/600 Robot, Volume 1: Technical Manual*, April 1980.

[21] R. Vistnes. Breaking away from VAL. Technical report, Unimation Inc., 1981.

[22] R.E. Walpole and R.H. Myers. *Probability and Statistics for Engineers and Scientists*. Collier Macmillan, 1978.