AIM104-MOTION-1

Technical Manual

Product Information

Full information about other Arcom products is available via the **Fax-on-Demand System**, (Telephone Numbers are listed below), or by contacting our **WebSite** in the UK at: **www.arcom.co.uk** or in the US at: **www.arcomcontrols.com**

Useful Contact Information

Customer Support

Tel:	+44 (0)1223 412 428
Fax:	+44 (0)1223 403 409
E-mail:	support@arcom.co.uk

Sales

Tel:	+44 (0)1223 411 200
Fax:	+44 (0)1223 410 457
E-mail	sales@arcom.co.uk

or for the US: E-mail
icpsales@arcomcontrols.com

United Kingdom
Arcom Control Systems Ltd
Clifton Road
Cambridge CB1 4WH, UK
Tel: 01223 411 200
Fax: 01223 410 457
FoD: 01223 240 600

United States
Arcom Control Systems Inc
13510 South Oak Street
Kansas City MO 64145 USA
Tel: 816 941 7025
Fax: 816 941 0343
FoD: 800 747 1097

France
Arcom Control Systems
Centre d'affaires SCALDY
23 rue Colbert
7885 SAINT QUENTIN
Cedex, FRANCE
Tel: 0800 90 84 06
Fax: 0800 90 84 12
FoD: 0800 90 23 80

Germany
Kostenlose Infoline:
Tel: 0130 824 511
Fax: 0130 824 512
FoD: 0130 860 449

Italy
Numero Verde:
Tel: 1677 90841
Fax: 1677 80841
FoD: 1678 73600

Belgium
Groen Nummer:
Tel: 0800 7 3192
Fax: 0800 7 3191

Netherlands
Gratis 0800 Nummer:
Tel: 0800 0221136
Fax: 0800 0221148

Preface

Packing List

This product is shipped as follows:

- Board
- Technical Manual
- Software Utility Disk

If any of the above appear to be missing, please telephone Arcom on 01223 411200.

Utility Disk

This product is shipped with a utility disk, 'Software for the AIM104-MOTION-1' v1.0. See the software section of this manual for disk content details.

Warning

This board contains **CMOS** devices which may be damaged by static electricity. Please ensure anti-static precautions are taken at all times when handling this board. If for any reason this board is returned to Arcom Control Systems, please ensure it is adequately packed to prevent damage during shipment.

Caution: Hot Surface

When using the on-board drivers, the two motor driver IC's (IC16 and IC17) may become very hot. Use caution when handling the board as they may remain hot for some time after power down.

Revision History

Manual	PCB	Comments	
Issue A	V1 Iss 1	980513	First full release of Manual
Issue B	V1 Iss 2	980720	[ECO2746].

Contents

Introduction

The **AIM104-MOTION-1** is an 8-bit PC/104 module providing single axis motor control, using Hewlett Packard's HCTL-1100 motion controller IC. The module may be used to directly drive a range of stepper motors or brushed DC servos up to a maximum of 1A per winding for closed loop operation (encoder feedback must be provided for independent motor control). Multiple boards can be connected together to provide synchronized multi-axis control.

Features

DC servo and stepper motors

- Single axis **brushed DC servo** or **stepper motor** control.
- On-board drivers. External motor supply of 12V-40V required.
- 1A max. output per winding .
- Single-ended or differential incremental quadrature encoder input (2 or 3 channel) up to 300KHz count rate. 24-bit position counter.
- 2 opto-isolated limit inputs, immediately halt motor.
- One opto-isolated stop input, motor halted with controlled deceleration.
- One opto-isolated home position input.
- Selectable motor current limit (via external resistor or link selectable).
- Programmable digital filter. Parameters changeable on-the-fly.
- Position control, velocity control (proportional and integral) and trapezoidal profile control for point to point moves.
- Jumper selectable I/O address (between 0H - 3FCH).
- Board access LED and external motor supply power-on LED.
- Interrupt on status change (limit, stop, profile complete) if desired. Shared interrupt option available. Jumper selectable interrupt line (IRQs 3,4,5,7,9,10,11,12,15).
- Operating temperature range, -20°C to +50°C (using on-board drivers) -20°C to +70°C (using external motor drivers)
- Power consumption from the PC/104 host, max. 350mA @ +5V.
- MTBF: 188,400 hours (using on-board motor drivers).
  405,000 hours (using external drivers).

Calculated using generic figures from MIL-HDBK-217F at ground benign.

Brushed DC servo motors

- 8-bit PWM control of brushed DC servo.
- PWM + Direction signals available for use by external drive amplifier.
- 2 or 3 channel incremental rotary or linear (quadrature) encoder should be used.

Stepper motors

- Full step (one phase-on and two phase-on modes) and half step operation supported.
- Driver commutator control signals available for external amplification.
- Stepper drive current output shaping for constant torque control in half-step mode (if stepper rated at 0.7A or less).
- 2 or 3 channel rotary incremental encoder should be used.

Operation

Control of the AIM104-MOTION-1 is achieved by downloading the motion information (e.g. target position, maximum velocity and acceleration) to the controller's internal registers.

The controller has a number of internal registers, each with a unique register address (see appendix 2 for register addresses and register details). Accessing the HCTL-1100 motion controller's registers is carried out by writing the register address to the base address (link selectable) and then accessing this register (read or write) at the base address + 1 location. (Further details given below).

Some AIM104-MOTION-1 settings are software selectable via the Configuration register. This is a directly addressable register at base address + 3.

There are also various board status flags which are accessable via the Status register. This is a directly addressable (read only) register located at base address + 2.

I/O Map

The board decodes 4 bytes of address space

Register map

Address	Function	Read/Write
Base address	Write register address to motion controller port	Write
Base address	Dummy read from motion controller port	Read
Base address + 1	Write data to motion controller port	Write
Base address + 1	Read data from motion controller port	Read
Base address + 2	Clear interrupt/reset status flags	Write
Base address + 2	Status register	Read
Base address + 3	Configuration register	Read/Write

Board Status register

Status Bit	Function
0	Ready Flag 0 = motion controller ready for Read/Write 1 = motion controller busy, wait
1	Limit 1 flag 0 = set (Limit triggered) 1 = cleared (no Limit)
2	Limit 2 flag 0 = set (Limit triggered) 1 = cleared (no Limit)
3	Stop flag 0 = set (Stop triggered) 1 = cleared (no Stop)
4	Home flag 0 = set (Home triggered) 1 = cleared (no Home)
5	Trapezoidal Profile flag 1 = in Profile Control
6	Unused
7	Unused

Note: Bits 1-5 of the status register remain set until cleared by writing any value to the 'Clear interrupt' address. If interrupts are not used and the Home and Limit 2 inputs are not used then the 'Clear interrupt' command is not necessary. The (latched) status of the Limit and Stop flags is available from the motion controller registers (see HCTL1100 data sheets) and these flags must be cleared separately.

Configuration register

Bit Number	Function
0	Motor drive bit 0 (see below)
1	Motor drive bit 1 (see below)
2	1 = Toggle SYNC line Automatically reset to 0
3	Unused
4	Unused
5	Unused
6	Unused
7	Unused

Motor drive bits

Bit 1	Bit 0	Function
0	0	DC servo motor
0	1	Stepper motor, full step, one phase on
1	0	Stepper motor, full step, two phases on
1	1	Stepper motor, half step mode

The motor drive bits determine the combinational logic circuit applied to the outputs of the controller, in order to provide the correct drive sequence for the motor type and mode selected.

Reading and Writing to the motion controller registers

The following procedures **must** be carried out in order to access the motion controller registers.

WRITING
1) Wait until board status bit 0 is not set
2) Write controller register address value to base address
3) Wait until board status bit 0 is not set (or delay >1.7µs)
4) Write data value to base address+1

READING
1) Wait until board status bit 0 is not set
2) Write controller register address value to base address
3) Dummy read from base address (this is essential)
4) Wait until board status bit 0 is not set (or delay >1.7µs)
5) Read data from base address+1

To operate the AIM104-MOTION-1 the following procedure is carried out:

1) The board configuration register is programmed for the appropriate motor type.
2) The motion controller IC is initialised.
3) The motion controller's digital filter parameters are set up.
4) The motor position is initialised.
5) The motor's maximum velocity and acceleration parameters are programmed (where required).
6) The motion controller is put into the required mode.
7) Motor position or velocity requirements are programmed as required.

Note: all parts 1-7 are required after a power-on. After that any part (1-7) can be carried out in any order if required. In simple motion control applications commonly only part 7 is repeated once the board has been fully initialised. Example of how to perform each of these functions are given in the example 'C' programs provided with this board.

If using a stepper motor with the AIM104-MOTION-1, it is important to read the 'Align Mode' section in Appendix 1.

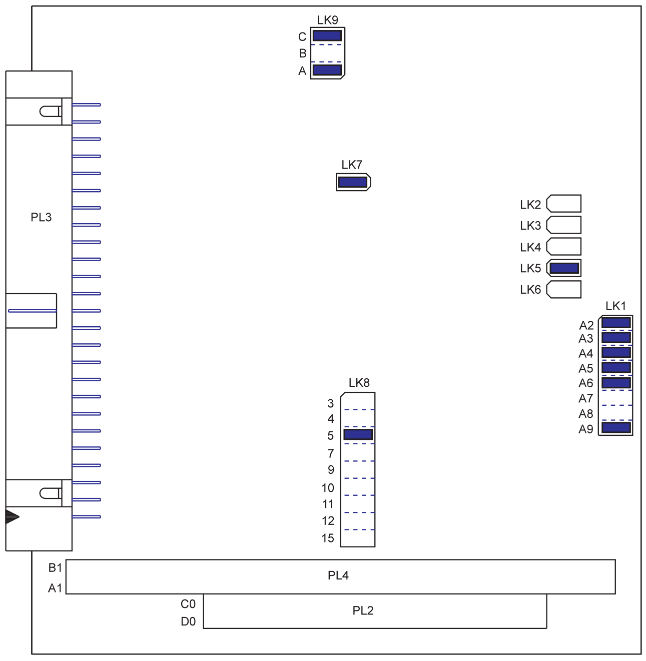
Further details on the HCTL-1100 IC's parameters are given in Hewlett Packard's data sheets 'General Purpose Motion Control ICs, Technical Data', available from Hewlett Packard electronic component distributors or from their internet site: 'http://www.hp.com/HP-COMP/motion/hctl1100.html'.

The HP application note, AN1032 'Design of the HCTL-1100's Digital Filter Parameters by the Combination Method' is also available from the above site.

Links

Throughout this section a '+' indicates a default link.

Default Link Position Diagram



Base Address Select

Links LK1A-G

Note: when a link is fitted the address is decoded as a '0' and when a link is omitted the address is decoded as a '1'.

The default address is set to 180H

Link	Address Line
+ LK1H	A9
LK1G	A8
LK1F	A7
+ LK1E	A6
+ LK1D	A5
+ LK1C	A4
+ LK1B	A3
+ LK1A	A2

Link 2

Default - off. Do not link.

Interrupt Share Option

Links 3 and 4

The AIM104-MOTION-1 is compatible with the PC/104 interrupt sharing option so that multiple interrupting devices can share a single bus interrupt line. By setting links 3 and 4 appropriately the module can be configured for shared interrupt line option or normal P996 bus operation.

P996 operation	- LK3 Open (default)
	- LK4 Open (default)
Interrupt sharing	- LK3 Made (but see below)
	- LK4 Made

Important:

All PC/104 devices sharing a common interrupt must be equipped with a suitable interrupt sharing circuit. If an interrupt line is to be shared by two or more devices then the line being shared must have one (and only one) pulldown resistor (1K ohms) connected between the IRQ line and ground. To do this on the AIM104-MOTION-1 link 3 should be inserted.

SYNC Select

Link 5 (default is link made)

The AIM104-MOTION-1 has a SYNC signal input which is used to reset the motion controller's timer. When link 5 is made the internal SYNC is selected. The SYNC signal can be pulsed by setting bit 2 of the AIM104-MOTION-1 configuration register. When synchronizing two or more AIM104-MOTION-1 boards it is desirable to synchronize all the timers. To do this, the SYNC lines of each board (pin 46 of PL3) should be tied together, and link 5 omitted on all boards except one.

HOME position halts motor

Link 6 (default is off)

When link 6 is made an active HOME input will halt the motor. When link 6 is made the HOME input acts like the LIMIT inputs but with one important difference. If a limit input halts the motor then the motor cannot be moved in the normal manner until the LIMIT signal is no longer active (see HCTL1100 data sheets for more information).

An active HOME signal however causes a short active pulse to be output to the motion controller's limit input. This ensures that the input to the motion controller is cleared regardless of whether the HOME input is still active, allowing the motor to be controlled as normal. The HOME signal must return high (inactive) and then low again (active) before it will re-trigger an active pulse to the motion controller.

Current Shaping Disable (only used with stepper motors)

Link 7

When link 7 is made, 'current shaping' mode is off. (Factory default is link made)

When driving a stepper motor in half-step mode there is usually a loss of torque in comparison to (2 phase-on) full step mode. This is because there is only one phase being driven during half of the cycle. However half-step mode gives much smoother speed-torque characteristics than full step mode.

In order to benefit from half-step mode without loss of torque, the board is capable of shaping the current drive to the motor, boosting the drive output by a factor of $\sqrt{2}$ during the time in which only one phase is active. This is permissible even if, according to the data sheet, the motor current limit has been reached, because this limit refers always to the contemporary supply with current in both windings in the full-step position. The resulting dynamic torque amounts to about 95% of that of the full-step torque.

Important: Driver, connector and wiring considerations mean that the maximum current drive per phase must be limited to 1A when using the on-board drivers. This means that current shaping must only be used when driving stepper motors with the current limit set to 0.7A or less. Current shaping will have no effect if used when driving DC servo motors.

Interrupt Select

Link 8

Link 8 determines which PC/104 IRQx line the AIM104-MOTION-1 interrupt will be generated upon. The default interrupt line is IRQ5.

Link	Interrupt Line
LK8A	IRQ3
LK8B	IRQ4
LK8C +	IRQ5
LK8D	IRQ7
LK8E	IRQ9
LK8F	IRQ10
LK8G	IRQ11
LK8H	IRQ12
LK8J	IRQ15

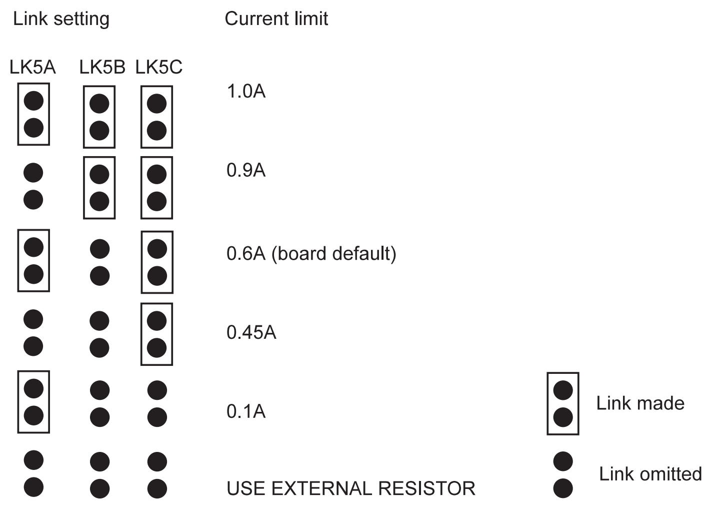
If an interrupt line has been selected, the AIM104-MOTION-1 will generate an interrupt whenever:

i) A LIMIT or STOP input has become active,
ii) A Trapezoidal Profile command has been completed, or
iii) The HOME input has become active AND Link 6 is made.

Current Limit Select

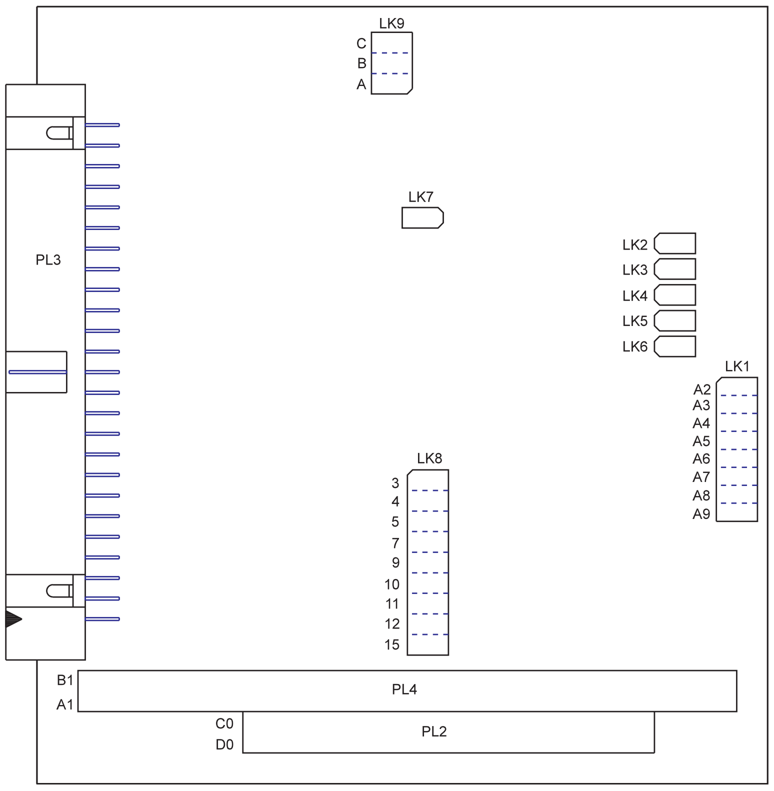
Link 9

Link 9 selects the current limit for each motor winding. If the on-board motor drivers are not used then link 9 has no effect. If an external resistor is to be used to limit the current then omit all 3 links from link 9 (and consult the table on page 12 for resistor value selection).



Note: If an external resistor is selected but is not connected, the current limit will be 0A.

User Configuration Diagram



Base Address Select	LK1	
Interrupt Share Option	LK3	
	LK4	
SYNC Select	LK5	
HOME position halts motor	LK6	
Current Shaping Disable	LK7	
Interrupt Select	LK8	

Connections and Pin Assignments

Connector (PL3) Pin Assignments

Pin No.	Signal Name	Pin No.	Signal Name
1	+Vs (motor supply)	26	B
2	+Vs (motor supply)	27	/B
3	+Vs (motor supply)	28	GND
4	GND (motor supply)	29	I
5	GND (motor supply)	30	/I
6	GND (motor supply)	31	GND
7	A1 ***40V, 1A max.***	32	+12V
8	A2 ***motor***	33	LIMIT 1
9	B1 ***drive***	34	GND
10	B2 ***outputs***	35	+12V
11	GND	36	LIMIT2
12	Phase A	37	GND
13	Phase B	38	+12V
14	Phase C	39	HOME
15	Phase D	40	GND
16	GND	41	+12V
17	PWM	42	STOP
18	GND	43	GND
19	Direction	44	External Limit Ref. (Ext. gnd or 12-24V)
20	GND	45	GND
21	+5V (for encoder power)	46	SYNC
22	GND	47	GND
23	A	48	Vres (for current limiting resistor)
24	/A	49	Current Limit
25	GND	50	GND

Connection of External Supply and Motor to the AIM104-MOTION-1

If the on-board drivers are not used then the appropriate control signals (PWM and Direction for DC servo motors, or Phases A,B,C and D for stepper motors) must be connected to a suitable amplifier module.

If the on-board drivers are to be used to drive a motor directly then an external DC power supply (12-40V) must be connected to the board. Ensure that the positive side of the supply is connected to **ALL** pins 1,2 and 3, and that the power supply ground is connected to **ALL** pins 4,5 and 6 (of PL3).

Brushed DC servo motors

Connect the motor between A1 and A2 (pins 7 and 8 of PL3).

Stepper motors

Stepper motors must be bipolar stepper motors. Unipolar stepper motors may be used if the windings are wired (in series or parallel) to be bipolar. It is recommended that where possible (unipolar motors with 8-wire configuration), the parallel configuration is used. If the motor has only six wires then it can only be wired in series configuration. See Appendix 4 for more details.

One phase of the stepper motor should be wired to A1 and A2 (pins 7 and 8 of PL3), and the other phase should be connected to B1 and B2 (pins 9 and 10 of PL3).

Important: If the motor is driving an inertial load then it must be ensured that either the power supply can **sink current** or that a suitably large capacitor with a high enough voltage rating (e.g. 1000µF, 63V) is connected across the power supply as the motor current is recirculated via the supply when the motor is braking. The capacitor should be connected as close to the board connector (PL3) as possible. **The power supply voltage should never be allowed to exceed 46V at any time.**

Incremental Encoder Input

A 2 or 3 channel incremental quadrature encoder must be connected to the board in order for the AIM104-MOTION-1 board to operate correctly. For stepper motors, best operation will be obtained by using a rotary encoder attached to the shaft of the motor. For DC motors a linear incremental encoder may be used instead where appropriate. Encoder power may be taken from pins 21 (+5V) and 22 (GND) of PL3. If a +12V supply is required then this can be taken from any of the +12V supply pins of PL3.

If a differential encoder is used connect A, /A, B and /B to PL4 pins 23, 24, 26 and 27 respectively. If 3 channels are available connect channel I to pins 29 (I) and 30 (/I). If the encoder is not differential then connect the signals (A, B and possibly I) to the positive channel inputs (pins 23, 26 and 29).

When choosing an encoder to use with a stepper motor, the user should keep in mind that the number of quadrature encoder counts (4x the number of slots in the encoder's codewheel) must be an integer multiple (1x, 2x , 3x, 4x etc.) of the number of steps of the stepper motor. In order to use the half-step mode of the board, the number of quadrature counts must be an even multiple (2x, 4x, 6x etc.) of the number of steps of the stepper motor.

The encoder input is used by the motion controller to determine the position of the motor. The HCTL-1100 has a 24-bit (2's complement) position counter. The position values are used directly when calculating the required distance to move when the motor is in any of its position modes. The number of counts received each sample period is used by the controller for determining the motor velocity.

Note: If the encoder inputs are incorrectly wired, operation of a motor in closed loop control will be unpredictable. The position loop can be tested before attempting closed loop operation by running the 'position' program (included in the Utilities disk) while the motor/encoder is connected.

External Current Limiting Resistor

If a current limit other than one that can be set using links 9A-9C is required, use the table below to determine the necessary external resistor value.

Resistor Value (Ohms)	Current Limit (Amps)	Resistor Value (Ohms)	Current Limit (Amps)
2700	1.00	9100	0.31
3000	0.91	10000	0.29
3300	0.83	11000	0.26
3600	0.77	12000	0.24
3900	0.71	13000	0.22
4300	0.65	16000	0.18
4700	0.60	18000	0.16
5100	0.55	20000	0.14
5600	0.50	24000	0.14
6200	0.46	30000	0.10
6800	0.42	36000	0.08
7500	0.38	47000	0.06
8200	0.35	56000	0.05

The external limiting resistor should be connected between pins 48 and 49 of PL3. All Link 9 links (A-C) must be removed if an external resistor is used.

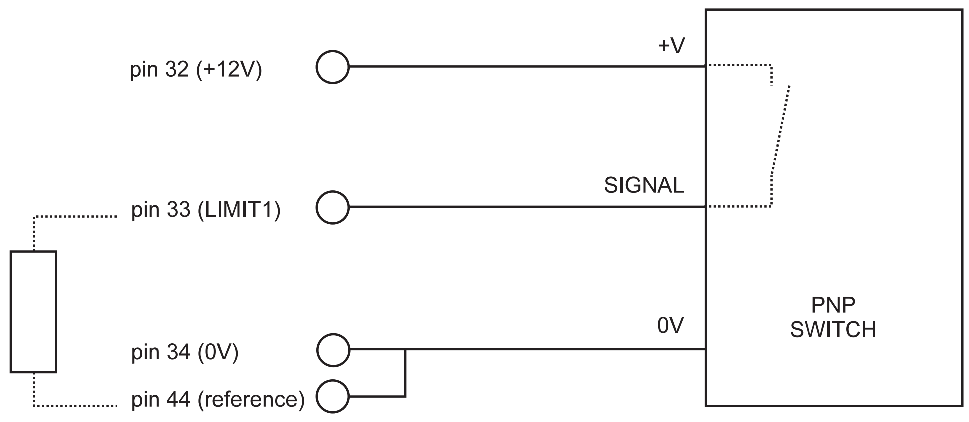
Limit Signal Inputs

The LIMIT1, LIMIT2, STOP and HOME inputs are all opto-isolated. An open circuit between the signal input pin and the ‘external limit reference’ input pin is the normal inactive state. A voltage difference between the signal input pin and the ‘external limit reference’ input (of between 5V and 24V) is the signal active state. The reference input may be positive or negative with respect to the signal input because AC opto-isolators are used. Thus either NPN or PNP switches may be used. If power for the limit switches is required, +12V and GND are present on the 50-way connector (PL3) for this purpose. +12V is only available if supplied to the board on the PC/104 bus.

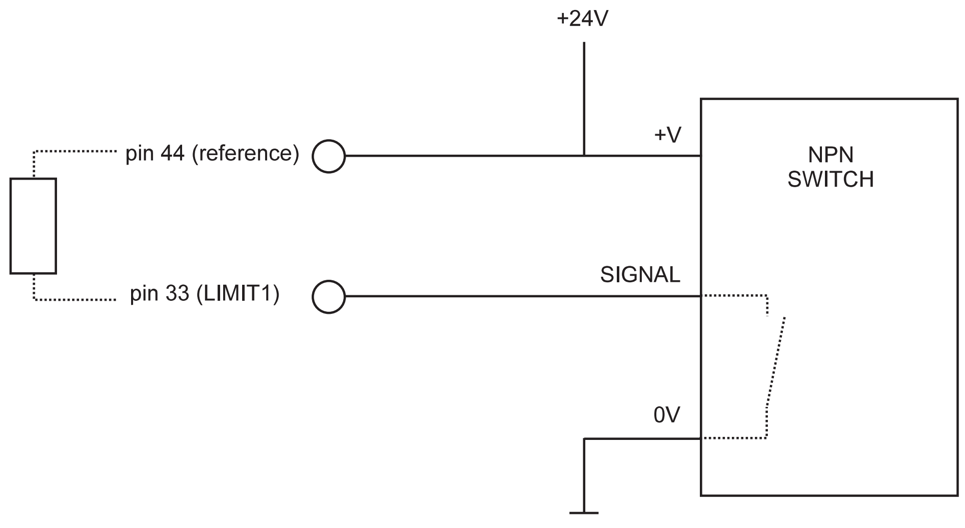
Important: All limit switch inputs must use the same reference input, which must be connected to pin 44 of PL3, even if the +12V supply from the AIM104-MOTION-1 is used for the switches. The +12V supply should not be used for purposes other than powering limit switches or rotary encoders.

e.g.

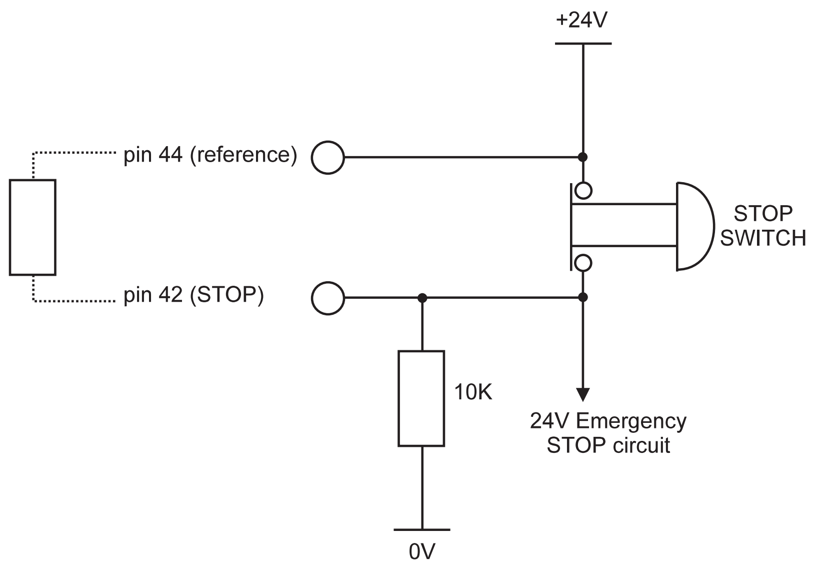
PNP (using AIM104-MOTION-1 power)



NPN (using external power)



Emergency STOP Switch



EMC Issues

The AIM104-MOTION-1 includes additional filter components on board to minimise the emissions of high frequency noise. The filters require that the earth tab supplied with the module is connected by a good earth wire to the chassis of the system.

The AIM104-MOTION-1 is classified as a ‘component’ with regard to the European Community EMC regulations and it is the user’s responsibilty to ensure that the systems using the board are compliant with the appropriate EMC standards.

In order to ensure that the board is protected from external interference and to minimise emmissions from the board it is strongly advised that:

1) Both the encoder (feedback) cable and the motor drive cable should be fully screened.
2) All cables should be kept as short as possible. Where feasible, keep motor and encoder cables less than 1m long.
3) Motor and encoder cables should not be routed together.
4) The cable screen should **NOT** be connected to signal ground at any point.

The driver ouputs may be filtered using a suitable LC type filter. However it is important that the filter capacitor is small enough in value (e.g. 200pF) so that the peak current (when the driver is switching) does not exceed the current limit (set by Link 9). If the capacitance is too large, the peak current will exceed the set limit and thus the current limiting circuit will continually be switching the drive output off, and correct operation will not be possible.

Software for the AIM104-MOTION-1

Introduction

In order to support this card a software utitlies disk is provided. The disk contains several ‘C’ source code example programs, a ‘C’ header file defining a large number of useful routines for easy communication with and control of the AIM104-MOTION-1, as well as a DOS based program to help calculate the correct values for the registers of the motion controller.

This software support has been developed to support Borland C++ versions 3.1 and 4.52, running on 386, 486 and Pentium CPU’s and Arcom Target boards. Full source code is provided to permit adaptation to other compilers.

Installation

To make use of the Utilities disk you will need either:

- A PC AT compatible, 386sx 25MHz or better, 1Mbyte DRAM (4Mbytes recommended), and 1.5Mbytes free hard disk space for software installation. Note that your particular compiler may well need a higher specification PC than the utilities software does to run satisfactorily. See your compiler documentation for details.

or:

- An Arcom SBC (single board computer) or Arcom Target Board.

To run any of the example programs you will need to fit the AIM104-MOTION-1 card to the Target board or to an ‘ISA to PC/104 bus adaptor’ card if there are no PC/104 connectors on the motherboard.

The software environment you will need is as follows:-

- MS-DOS ver 5.0 or later
- BorlandC++ version 3.1 or 4.52 (if you plan to compile any of the example programs and/or make use of the header file).

To install: Place the Utilities disk in the A drive. At the DOS prompt type A:\install. If you are running Windows 95 either open up a DOS window or you can use Windows Explorer to copy the files across to the C drive.

The files will be installed on the hard disk in the C:\MOTION directory. If you wish to install the files elsewhere, then the destination directory may be changed by altering the INSTALL.BAT file in the root directory of the Utilities disk.

What’s on the Disk?

The installation program installs the following files:-

```
C:\
└──────MOTION                 Root
        ├──────────EXAMPLES   Example files directory
        ├──────────DOCS       Documentation files directory
        ├──────────INCLUDE    Include files directory
        └──────────REGISTER   Register value calculation help program
```

The Examples directory (EXAMPLES) should contain:-

STEPPER1.C	Position Control routine for stepper motor in 1/2 step mode.
STEPPER2.C	Trapezoidal Profile control routine for stepper motor in 1/2 step mode.
STEPPER3.C	Proportional Velocity control routine for stepper motor in 1/2 step mode.
STEPPER4.C	Integral Velocity control routine for stepper motor in 1/2 step mode.
STEPPER5.C	Position Control routine for stepper motor in full step mode.
STEPPER6.C	Open-loop control of stepper motor in 1/2 step mode.
DCSERVO1.C	Position Control routine for DC servo motor.
DCSERVO2.C	Trapezoidal Profile control routine for DC servo motor.
DCSERVO3.C	Proportional Velocity control routine for DC servo motor.
DCSERVO4.C	Integral Velocity control routine for DC servo motor.
DCSERVO5.C	Example interrupt routine for DC servo motor.
POSITION.C	Test program to check the encoder feedback loop.

For Borland C++ 4.52 Users:-

STEPPERx.IDE	Project files for Borland integrated development environment.
POSITION.IDE	
DCSERVOx.IDE	

The include files directory (INCLUDE) should contain:-

GLOBALS.H	General Arcom PC/104 definitions
MOTION.H	Motion function prototypes

The documentation directory (DOCS) should contain:-

CONTENTS.TXT	Detailed list of disk contents
MOTION.TXT	Description of all functions included in MOTION.H

The register help program directory (REGISTER) should contain:-

REGISTER.EXE	DOS program to help calculate the motion controller register values required.

Library Functions

To make use of the motion controller functions you will need to include the file 'motion.h' in all of the modules which use any of the functions listed below.

The first function you need to call in all programs which use the AIM104-MOTION-1 board is MOTION_boardinit(unsigned char baseaddr) to which you will pass the base address of the board. This function returns a 'handle' which you then use in all other ‘motion.h’ function calls.

MOTION_boardinit() is very similar to the COMMON_boardinit() function included in the AIM104 Software Library. MOTION_boardinit() will return a unique handle value not used by any board initialised with COMMON_boardinit(). Note that COMMON_boardinit() should not be used to initialise the AIM104-MOTION-1 board.

Please read Appendix 2, regarding the HCTL-1100 registers for details of the units used when programming the registers. Some registers are one byte in size, some two and some are three bytes. E.g. MOTION_set_max_velocity() uses one byte, with the units of quadrature counts/sample period. Whereas, the function MOTION_set_proportional_velocity() uses two bytes (i.e. it requires an input of ±32768), with its units being 1/16 of a quadrature count/sample period.

The following is a summary of the functions, their parameters and their usage.
Refer to the example programs for help on how to use the functions.

`int MOTION_boardinit(unsigned short baseaddr)`
- Initialises AIM104-MOTION-1 board

`int MOTION_boardclose(int handle)`
- De-initialises AIM104-MOTION-1 board

`MOTION_wait_until_ready(int handle)`
- Waits until motion controller is ready for the next read/write access, before returning

`MOTION_set_board_config_reg(int handle, char config_value)`
- Writes a byte value to the boards configuration register

`char MOTION_read_board_config_reg(int handle)`
- Returns the byte value of the boards configuration register

`MOTION_pulse_SYNC(int handle)`
- Forces a SYNC pulse to be generated, resetting the boards Sample Timer

`MOTION_set_controller_status_reg(int handle, char status)`
- Writes a byte value to the motion controller's status register

`MOTION_program_sample_timer(int handle, char period)`
- Writes the timing period to the Sample Timer register (in units of 8us)

```
MOTION_set_filter_parameters(int handle,char filter_zero,
                   char filter_pole,char filter_gain)
```
- Sets the motion controller's digital filter parameters

`MOTION_controller_reset(int handle)`
- Causes a software reset of the motion controller IC

`MOTION_clear_interrupts(int handle)`
- Clears the interrupt line and the interrupt flags of both the board's status register and the motion controller's status register

`MOTION_enter_idle_mode(int handle)`
- Forces the board into the init/idle mode

`MOTION_enter_align_mode(int handle)`
- Forces the board into the align mode

`MOTION_enter_position_mode(int handle)`
- Forces the board into the position mode

`MOTION_execute_trapezoidal_move(int handle)`
- Starts a trapezoidal move, as defined by MOTION_set_final_position, MOTION_set_acceleration and MOTION_set_max_velocity

`MOTION_execute_proportional_velocity(int handle)`
- Starts a motor in proportional velocity mode, the velocity is set by MOTION_set_proportional_velocity

`MOTION_end_proportional_velocity(int handle)`
- Stops the proportional velocity move, leaving the board in position mode

`MOTION_execute_integral_velocity(int handle)`
- Starts a motor in integral velocity mode, as defined by MOTION_set_integral_velocity and MOTION_set_acceleration

`MOTION_end_integral_velocity(int handle)`
- Ends integral velocity profile, leaving the board in position mode

`char MOTION_wait_until_move_complete(int handle)`
- Waits until a trapezoidal proile move has completed, returning a zero on a successful completion

`MOTION_set_command_position(int handle,unsigned long command_position)`
- Sets the command position for position mode (command position is also required in trapezoidal profile mode as well)

`MOTION_set_final_position(int handle,unsigned long final_position)`
- Sets the desired final position in a trapezoidal move

`MOTION_preset_actual_position(int handle,unsigned long actual_position)`
- Presets the current position by setting the encoder position counter

`MOTION_set_acceleration(int handle, int accel)`
- Sets the acceleration for the integral velocity and trapezoidal profile modes

`MOTION_set_max_velocity(int handle, char velocity)`
- Sets the maximum allowed velocity in the trapezoidal profile mode

`MOTION_set_proportional_velocity(int handle, int velocity)`
- Sets the target velocity for the proportional velocity mode

`MOTION_set_integral_velocity(int handle, char velocity)`
- Sets the velocity for the integral velocity mode

```
MOTION_program_commutator(int handle,char velocity_timer,char ring,
                 char X,char Y,char offset,char max_phase_advance)
```
- Programs the HCTL-1100 motion controller's commutator parameters

`char MOTION_read_board_status_reg(int handle)`
- Returns the value of the board's status register

`char MOTION_read_controller_status_reg(int handle)`
- Returns the value of the motion controller’s status register

`char MOTION_read_flag_register(int handle)`
- Returns the flag settings of the motion controller

`unsigned long int MOTION_read_command_position(int handle)`
- Returns the command position register value

`unsigned long int MOTION_read_final_position(int handle)`
- Returns the value last programmed into the final position register

`unsigned long int MOTION_read_actual_position(int handle)`
- Returns the actual position register contents

`int MOTION_read_actual_velocity(int handle)`
- Returns the actual velocity as calculated by the motion controller when in the proportional velocity mode

`unsigned char MOTION_read_sample_timer(int handle)`
- Returns the current sample timer count value.

```
char  MOTION_open_loop_half_stepper(int  handle,int  total_steps,int
                 step_period)
```
- Function to control a stepper motor in open loop mode (no encoder feedback) - using this function ties up the host processor while the move is in operation

APPENDIX 1- HCTL-1100 Modes

There are three setup routines and four control modes of the HCTL-1100 IC. The setup routines are:

Reset
Initialization/Idle
Align

The four control modes are:

Position Control Mode
Trapezoidal Profile Control Mode
Integral Velocity Mode
Proportional Velocity Mode

Each mode is explained in more detail below.

Reset

Upon power-up a hard reset is executed:

All Flags (F0-F5) are cleared.
The commutator logic is cleared.
The I/O logic is cleared.
The pulse pin is set low. At the end of the hard reset the pulse pin goes high for a period of 0.5 us.

At the end of the hard reset a soft reset is automatically executed. A soft reset is also executed whenever 00H is written to Register 05H (the Program Counter Register). When a soft reset is executed the following conditions occur:

The Actual Position Counters are set to 0.
The digital filter parameters are set to: A = E5H, B = 40H, K = 40H
The Sample Timer Register is preset to 40H.
The Status Register is cleared.

From the reset mode the HCTL-1100 automatically enters the Initialisation/Idle mode.

Initialization/Idle

The Initialization mode is executed after the reset routine, or when one of the Limit inputs goes low, or by writing 01H to Register 05H (the Program Counter Register).

In this mode the following occur:

The Init/Idle flag is set.
The PWM port (Register 09H) is set to 00H (zero output).
Previously sampled data in the digital filter is cleared.

It is in this mode that the user should pre-program all the necessary registers needed to execute the desired control mode. The HCTL-1100 will stay in this mode until a new mode is commanded.

Align

This mode is only used with the AIM104-MOTION-1 when using a stepper motor.

At first power on, the on-board controller does not know the phase orientation of the stepper motor. Without first aligning the motor to a known phase, accurate positioning and control of the motor is unpredictable. This is because the controller only advances to the next step phase once the encoder has advanced the expected number of counts.

The align mode can only be entered from the Initialisation/Idle mode, and is entered by writing 02H to the Program Counter Register (R05H). Before entering the align mode, all control mode flags should be cleared and both the Command Position Registers and the Actual Position Registers should be set to zero. The commutator parameters should also have been correctly programmed.

If the index output (channel I) of an encoder is used, it must have been correctly aligned to the last motor phase during motor/encoder assembly.

The align mode first disables the commutator and then using open loop control, enables the 1st phase (PHA) and then the last phase (PHD) to orientate the motor on the last phase detent. Each phase is energised for 2048 Sample Time periods.

At this point the commutator is re-enabled and the controller will enter the Position Control mode and go to position zero.

The Control Modes

The Control flags F0,F3, and F5 in the Flag Register (00H) determine which control mode is executed. Only one control flag can be set at one time. Once the flags are appropriately set for the required control mode, the mode is entered from the Initialization/Idle by writing 03H to the Program Counter (05H). Alternatively, the Position Control mode is entered automatically after completion of the Align routine.

Position Control Mode

Flags:	F0 Cleared
	F3 Cleared
	F5 Cleared

Position Control performs point-to-point position moves with no velocity profiling. The user specifies a 24-bit position command, which the controller compares to the 24-bit actual position. The position error is calculated, the full digital lead compensation is applied and the motor command is output.

The controller will remain position-locked at a destination until a new position command is given. Position is measured in encoder quadrature counts. See the sections on Command Position Registers and Actual Position Registers for more details on the operation of these registers.

The largest position move possible in Position Control mode is 7FFFFFH (8,388,607D) quadrature encoder counts.

Proportional Velocity Mode

Flags:	F0 Cleared
	F3 Set
	F5 Cleared

Proportional Velocity Control performs control of motor speed using only the gain factor, K, for compensation. The dynamic pole and zero lead compensation are not used. (See the section on the “Digital Filter” in Appendix 2 for more details of the filter). The units of velocity are quadrature counts/sample time.

The command velocity and actual velocity are two 16-bit two’s-complement words. The command velocity resides in registers 24H (MSB) and 23H (LSB). These registers are unlatched which means that the command velocity will change to a new velocity as soon as the value in either R23H or R24H is changed. The registers can be read or written to in any order.

The command velocity is internally interpreted as 12 bits of integer and 4 bits of fraction (the lower nibble of R23H). Thus, the host processor must multiply the desired command velocity (in quadrature counts/sample time) by sixteen before programming into the HCTL-1100 registers.

The actual velocity is computed only when in Proportional Velocity mode. It is stored in registers 35H (MSB) and 34H (LSB). There is no fractional component in the actual velocity registers. The controller tracks the command velocity continuously until a new mode command is given.

To convert from rpm to quadrature counts/sample time, use the following formula:

$$Vq = (Vr)\ (N)\ (t)\ (0.01667) \qquad [1]$$

Where:	Vq	= velocity in quadrature counts/sample time
	Vr	= velocity in rpm
	N	= the number of quadrature counts per 1 revolution
	t	= the HCTL-1100 sample time in seconds.

Integral Velocity Mode

Flags: F0 Cleared
F3 Cleared
F5 Set

Integral Velocity Control performs continuous velocity profiling which is specified by a command velocity and a command acceleration. The user can change velocity and acceleration any time to continuously profile velocity in time. Once the specified velocity is reached, the HCTL-1100 will maintain that velocity until a new command is specified. Changes between actual velocities occur at the presently specified linear acceleration.

The command velocity is an 8-bit two's-complement word stored in register 3CH. The units of velocity are quadrature counts/sample time. The conversion from rpm to quadrature counts/sample time is shown in equation [1]. The command velocity register contains no fractional component.

Although the overall range of the command velocity is 8 bits, the largest difference between any two sequential command velocities cannot be greater than 7 bits in magnitude (i.e. 127 in decimal).

The command acceleration is a 16-bit scalar word stored in registers 27H and 26H. The upper register (27H) contains the integer part and the lower register (26H) contains the fractional part. The contents of register 26H are divided by 256 to produce the fractional resolution.

The units of acceleration are quadrature counts/sample time squared. To convert from rpm/sec to quadrature counts/sample time squared, use the following formula:

$$Aq = (Ar)\ (N)\ (t2)\ (0.01667) \qquad [2]$$

Where:	Aq	= Acceleration in quadrature counts/[sample time]^2
	Ar	= Acceleration in rpm/sec
	N	= the number of quadrature counts per 1 revolution
	t	= the HCTL-1100 sample time in seconds.

Internally the controller performs velocity profiling through position control. Each sample period, the internal profile generator uses the information which the user has programmed into the Command Velocity register (3CH) and the Command Acceleration registers (27H and 26H) to determine the value which will be automatically loaded into the Command Position registers. In control theory terms, integral compensation has been added (see HCTL-1100 data sheet for more information) and therefore, the system has zero steady state error.

Although Integral Velocity Control mode has the advantage over Proportional Velocity mode of zero steady state error, its disadvantage is that the closed loop stability is more difficult to achieve.

If the external Stop flag (F6) is set during this mode, signalling an emergency condition, the controller automatically decelerates to zero velocity at the presently specified acceleration factor and stays in this condition until the flag is cleared.

Trapezoidal Profile Mode

Flags:	F0 Set
	F3 Cleared
	F5 Cleared

Trapezoidal Profile Control performs point-to-point position moves and profiles the velocity trajectory to a trapezoid or triangle. The user specifies only the desired position, acceleration and maximum velocity. The controller computes the necessary profile to conform to the command data. If the maximum velocity is reached before the halfway point, the profile will be trapezoidal, otherwise the profile will be triangular.

The command data for Trapezoidal Profile Control mode consists of a final position, a command acceleration, and a maximum velocity. The 24-bit, two's-complement Final Position is written to registers 2BH (MSB), 2AH and 29H (LSB). The 16-bit command acceleration resides in registers 27H (MSB) and 26H (LSB). The command acceleration has the same integer and fraction format as discussed in the Integral Velocity Control mode section. The 7-bit maximum velocity is a scalar value with the range of 00H to 7FH. The maximum velocity has the units of quadrature counts per sample time, and resides in register 28H. The command data registers may be read or written to in any order.

The internal profile generator produces a position profile using the present Command Position (NOT Actual Position) as the starting point and the Final Position as the end point. Once the desired data is entered, the user sets flag F0 in the Flag register (00H) to commence motion (if the HCTL-1100 is already in Position Control mode).

When the profile generator sends the last position command to the Command Position registers to complete the trapezoidal move, the controller clears flag F0. The HCTL-1100 then automatically goes to Position Control mode with the final position of the trapezoidal move as the command position.

The status of the Profile flag can be monitored both in the HCTL-1100 status register (07H) and from the AIM104-MOTION-1 boards status register, bit 5. While the controller is in Profile NO new command data should be sent to the controller. Completion of a profile command does not indicate that the motor and encoder are at the final position NOR that the motor and encoder have stopped, only that the command profile has finished. The motor's true position can only be determined by reading the Actual Position registers.

APPENDIX 2- HCTL-1100 Registers

This section describes in brief the HCTL-1100 registers that may be accessed via the AIM104-MOTION-1 board. For a more detailed description please read the HCTL-1100 data sheets.

Register Reference Table

Register	Function	Mode Used	Data Type	User Access
R00H	Flag Register	All	–	r/w
R05H	Program Counter	All	scalar	w
R07H	Status Register	All	–	r/w[1]
R09H	PWM Motor Command Port	All	2's complement	r/w
R0CH	Command Position (MSB)	All, except Proportional Velocity	2's complement	r/w[2]
R0DH	Command Position	All, except Proportional Velocity	2's complement	r/w[2]
R0EH	Command Position (LSB)	All, except Proportional Velocity	2's complement	r/w[2]
R0FH	Sample Timer	All	scalar	r/w
R12H	Read Actual Position (MSB)	All	2's complement	r[3]
R13H	Read Actual Position	All	2's complement	r[3]/w[4]
R14H	Read Actual Position (LSB)	All	2's complement	r[3]
R15H	Preset Actual Position (MSB)	INIT/IDLE	2's complement	w[6]
R16H	Preset Actual Position	INIT/IDLE	2's complement	w[6]
R17H	Preset Actual Position (LSB)	INIT/IDLE	2's complement	w[6]
R18H	Commutator Ring	All	scalar[5]	r/w
R19H	Commutator Velocity Timer	Not Used	scalar	w
R1AH	X	All	scalar[5]	r/w
R1BH	Y Phase Overlap	All	scalar[5]	r/w
R1CH	Offset	All	2's complement	r/w
R1FH	Phase Advance	Not Used	scalar[5]	r/w
R20H	Filter Zero, A	All, except Proportional Velocity	scalar	r/w
R21H	Filter Pole, B	All, except Proportional Velocity	scalar	r/w
R22H	Gain, K	All	scalar	r/w
R23H	Command Velocity (LSB)	Proportional Velocity	2's complement	r/w
R24H	Command Velocity (MSB)	Proportional Velocity	2's complement	r/w
R26H	Acceleration (LSB)	Integral Velocity and Trapezoidal Profile	scalar	r/w
R27H	Acceleration (MSB)	Integral Velocity and Trapezoidal Profile	scalar[5]	r/w
R28H	Maximum Velocity	Trapezoidal Profile	scalar[5]	r/w
R29H	Final Position (LSB)	Trapezoidal Profile	2's complement	r/w
R2AH	Final Position	Trapezoidal Profile	2's complement	r/w
R2BH	Final Position (MSB)	Trapezoidal Profile	2's complement	r/w
R34H	Actual Velocity (LSB)	Proportional Velocity	2's complement	r
R35H	Actual Velocity (MSB)	Proportional Velocity	2's complement	r
R3CH	Command Velocity	Integral Velocity	2's complement	r/w

[1] Upper 4 bits are read only.
[2] Writing to R0EH (LSB) latches all 24 bits.
[3] Reading R14H (LSB) latches data in R12H and R13H.
[4] Writing to R13H clears Actual Position Counter to zero.
[5] The scalar data is limited to positive numbers (00H to 7FH).
[6] Writing to R17H latches all 24 bits (only in INIT/IDLE mode).

GENERAL REGISTER DESCRIPTIONS

FLAG REGISTER (R00H)

The Flag register contains flags F0 through F5. Each flag is set and cleared by writing an 8-bit data word to R00H. When writing to R00H, the upper four bits are ignored by the HCTL-1100. Bits 0,1,2 specify the flag address, and bit 3 specifies whether to set or clear the addressed flag. See the table below:

Flag	SET	CLEAR
F0	08H	00H
F1	—	—
F2	0AH	02H
F3	0BH	03H
F4	0CH	04H
F5	0DH	05H

When reading from the Flag Register, the status of the flags is returned in bits 0 to 5. For example, if bit 4 is set, then flag F4 is set. Bits 6-7 are undefined. See the table below:

Bit Number	Flag (1 = set) (0 = clear)
7-6	Don't Care
5	F5
4	F4
3	F3
2	F2
1	F1
0	F0

FLAG DESCRIPTIONS

F0 - Trapezoidal Profile Flag, set by the user to execute Trapezoidal Profile Control. The flag is reset by the controller when the move is completed.

F1 - Initialization/Idle Flag, set/cleared by the HCTL-1100 to indicate execution of the Initialization/Idle mode. The user should not attempt to set or clear F1.

F2 - Unipolar Flag. Not used by the AIM104-MOTION-1.

F3 - Proportional Velocity Control Flag, set by the user to specify Proportional Velocity control.

F4 - Hold Commutator Flag, set/cleared by the user or automatically by the align mode. When set, this flag inhibits the internal commutator counters, allowing open loop stepping of a motor by using the commutator.

F5 - Integral Velocity Control, set by the user to specify Integral Velocity Control.

PROGRAM COUNTER REGISTER (R05H)

The Program Counter, which is write only, executes the pre-programmed functions of the controller. The program counter is used along with the control flags F0, F3, and F5 in the flag register (R00H) to change control modes. The user can write any of the following four commands to the Program Counter.

00H	Software Reset.
01H	Enter Init/Idle Mode.
02H	Enter Align Mode (only from Init/Idle mode)
03H	Enter Control Mode (only from Init/Idle mode).

See appendix A, HCTL-1100 Modes.

STATUS REGISTER (R07H)

The Status Register indicates the status of the HCTL-1100. Each bit decodes into one signal. All 8 bits are user readable and are decoded as shown below. Only the lower 4 bits can be written to by the user to configure the HCTL-1100. To set or clear any of the lower 4 bits, the user writes an 8 bit word to R07H. The upper 4 bits are ignored.

Status Bit	Function
0	PWM Sign Reversal Inhibit 0 = off 1 = on
1	Commutator Phase Configuration (0 = 3 phase) *not used with AIM104-MOTION-1* 1 = 4 phase
2	Commutator Count Configuration 0 = quadrature 1 = full
3	Should always be set to 0
4	Trapezoidal Profile flag F0 1 = in Profile Control
5	Initialization/Idle Flag F1 1 = in Initialization/Idle Mode
6	Stop Flag 0 = set (Stop triggered) 1 = cleared (no Stop)
7	Limit Flag 0 = set (one of the 2 limits has triggered) 1 = cleared (no Limit)

PWM MOTOR COMMAND REGISTER (R09H)

The PWM port outputs the motor command as a pulse width modulated signal with the correct sign of polarity. The PWM port consists of the Pulse and Sign pins (PL3 pins 17 and 19 respectively) and Register 09H.

The 2's complement contents of R09H determine the duty cycle and polarity of the PWM command. The PWM signal at the Pulse pin has a frequency of 20KHz, and the duty cycle is resolved into 100 parts (e.g. writing '50' decimal to the PWM port would result in 20KHz signal with a duty cycle of 50%). The Sign pin gives the polarity of the command. Low output on the Sign pin is positive polarity.

The PWM port is only written to if moving a motor outside of closed loop control, e.g. to move off a limit switch. When a motor is being driven under the control of the HCTL-1100 in one of its programmed Control Modes, the PWM port is automatically controlled by the HCTL-1100. Thus the user should only write to R09H when the controller is in the Initialization/Idle mode.

The PWM (pulse) and Direction (Sign) pins are used if a DC servo motor is to be driven via an external amplifier. If driving a stepper motor via an external amplifier, only the PWM signal should be used (along with the Phase A-D commutator signals) as the commutator signals already contain direction information.

ACTUAL POSITION REGISTERS

Read, Clear:	R12H, R13H, R14H
Preset:	R15H, R16H, R17H

The Actual Position Register is accessed by two sets of registers in the HCTL-1100. When reading the Actual Position, read registers R12H (MSB), R13H, and R14H (LSB). When presetting the Actual Position, write to registers R15H (MSB), R16H, and R17H (LSB).

When reading the Actual Position, the order should be LSB first. When the LSB is read, all three bytes are latched so that the count data will not change while reading the three bytes.

When presetting the Actual Position, write to the LSB last. When the LSB is written to, all three bytes are simultaneously loaded into the Actual Position Register. Note that presetting the Actual Position is only allowed in the Init/Idle mode. The Actual Position registers can be simultaneously cleared at any time by writing to R13H.

DIGITAL FILTER REGISTERS

Zero (A)	R20H
Pole (B)	R21H
Gain (K)	R22H

All control modes use some part of the programmable digital filter D(z) to compensate for closed loop system stability. The compensation D(z) has the form:

$$D(z)= \frac{K\left(z-\frac{A}{256}\right)}{4\left(z+\frac{B}{256}\right)} \qquad [3]$$

where:

z	= the digital domain operator
K	= digital filter gain (R22H)
A	= digital filter zero (R20H)
B	= digital filter pole (R21H)

The compensation is a first-order lead filter which in combination with the Sample Timer T (R0FH) affects the dynamic step response and stability of the control system. The Sample Timer, T, determines the rate at which the control algorithm gets executed. All parameters, A, B, K and T are 8 bit scalars that can be changed at any time by the user.

In Proportional Velocity Control, the digital compensation filter is implemented in the time domain as:

$$MC_n = (K/4)(Y_n) \qquad [4]$$

where:

MC_n	= Motor Command Output at n
Y_n	= (Command Velocity - Actual Velocity) at n
n	= current sample time.

In Position Control, Integral Velocity Control and Trapezoidal Profile Control the digital filter is implemented in the time domain as shown below:

$$MCn = (K/4)(X_n) - [(A/256)(K/4)(X_{n-1}) + (B/256)(MCn-1)] \qquad [5]$$

where:

n	= current sample time
n-1	= previous sample time
MC_n	= Motor Command Output at n
MC_{n-1}	= Motor Command Output at n-1
X_n	= (Command Position - Actual Position) at n
X_{n-1}	= (Command Position - Actual Position) at n-1

For more information on system sampling times, bandwidth and stability please consult Hewlett Packard Application Notes ABM005 ‘Sample Timer and Digital Filter’ and AN1032 ‘Design of the HCTL-1000's/1100's Digital Filter Parameters by the Combination Method’.

SAMPLE TIMER REGISTER (R0FH)

The contents of this register set the sampling period of the HCTL-1100. The sampling period is:

$$t=\frac{16x(T+1)}{2x10^6} \qquad [6]$$

where: T= contents of register 0FH, the Sample Timer Register.

The Sample Timer has a limit on the minimum allowable sample time depending on the control mode being executed. The limits are given in the following table:

Control Mode	Reg. 0FH Contents Minimum Limit
Position Control	07H
Proportional Velocity Control	07H
Trapezoidal Profile Control	0FH
Integral Velocity Mode	0FH

The minimum values are to make sure the internal programs have enough time to complete proper execution.

The maximum value of T is FFH, thus the sample time can vary from 64µs to 2048µsec. Digital closed-loop systems with slow sampling times have lower stability and a lower bandwidth than similar systems with faster sampling times. To keep the system stability and bandwidth as high as possible the AIM104-MOTION-1 should be programmed with the fastest possible sampling time. However this must be balanced by the needs of the velocity range to be controlled. Velocities are specified in terms of quadrature counts per sample time. Where slow speeds need to be controlled, the sample period may need to be higher than the minimum allowed.

The Sample Timer consists of a buffer and a decrement counter. Each time the counter reaches zero, the Sample Timer Value T is loaded from the buffer into the counter, which immediately begins to count down. Data written to the Sample Timer (R0FH) will be latched into the internal buffer, and then used by the counter after it completes the present sample time cycle.

Reading from R0FH gives the values directly from the decrementing counter. Therefore, the data read will have a value anywhere between T and 00H, depending where in the sample time cycle the counter is.

COMMUTATOR REGISTER DESCRIPTIONS

The commutator registers are:

Commutator Ring	(R18H)
X Register	(R1AH)
Y Phase Overlap	(R1BH)
Phase Offset	(R1CH)
Velocity Timer	(R19H)
Max. Phase Advance	(R1FH)

The commutator is a digital state machine that is configured by the user to properly select the phase sequence for driving a stepper motor. It is not used when driving DC servo motors. Phase overlap can be programmed and this is used when driving in the half-step mode. With the AIM104-MOTION-1 board, for full-step modes, the phase overlap (Y) is always zero. For half-step modes, the phase overlap should always be equal to the X register value.

The inputs to the commutator are the encoders signals, channel A, channel B and Index (optional), as well as the configuration data stored in the commutator registers.

If the index channel of the encoder is used, it must be physically aligned to a known torque curve location (last phase on) because it is used as the reference point of the rotor position with respect to the commutator phase. The Phase Offset register is used as the fine adjustment for alignment, once the complete control system is set up.

The **Ring Register** is scalar and determines the length of the commutation cycle measured in full or quadrature counts (see Status register). The value of the ring must be limited to the range of 0 to 7FH. The Ring register value can be calculated as follows:

$$\text{Ring} = N / (\text{No. of steps/revolution} \div 4) \qquad [7]$$

where: N = number of full or quadrature encoder counts per revolution.

Each time an index pulse occurs, the internal commutator ring counter is reset to 0. The ring counter keeps track of the current position of the rotor based on the encoder feedback. When the ring counter is reset to 0, the commutator is reset to its origin.

Note: the commutator counting method can be set to use either full encoder counts or quadrature encoder counts by setting or clearing Bit 2 of the Status Register. In most control situations it would be programmed to use quadrature encoder counts. However, where the number of encoder counts per revolution is much larger than the number of steps per revolution it may be necessary to use only full counts, in order to satisfy the following equation:

$$-128D <= (3/2\ \text{Ring} + \text{Offset}) <= 127D \qquad [8]$$

Changing the commutator counting method to full counts only affects the commutator registers. Quadrature counts are always used as the basis for position, velocity and acceleration control.

The **X register** and **Y register** contain scalar data which is set as follows:

Stepper in Full Step mode: X = Ring register / 4 [9]
Stepper in Half Step mode: X = Y = Ring register / 8 [10]

In order to satisfy these equations and equation [7], it can be seen that number of quadrature encoder counts (4x the number of slots in the encoder's codewheel) must be an integer multiple (1x, 2x , 3x, 4x etc.) of the number of steps of the stepper motor. If Half Step mode is used then it must be an even integer multiple (2x, 4x, 6x etc.).

Note: The HCTL-1100 commutator output signals for both types of full-step mode ('one phase-on' and 'two phase-on') will be the same. The AIM104-MOTION-1 Configuration register bits 0 and 1 select the logic used by the AIM104-MOTION-1 to configure the stepper output waveform for the appropriate mode. Thus to switch between one full-step mode and the other only requires a write to the Configuration register, whereas switching between one of the full-step modes and half-step mode requires writing to the Configuration register, the X register and the Y register.

The **Phase Offset** value can be used to adjust the alignment of the commutator with the motor torque curves. The Offset register contains 2's complement data which determines the relative start of the commutation cycle with respect to the index pulse. In most cases the Phase Offset can be left at zero. Where maximum motor torque is required, this can be achieved by correctly aligning the commutator output with the motor's torque curves. Note, the encoder index channel should be present and correctly aligned to the last phase of the motor if using the phase offset. See also Phase Advance registers below.

The **Phase Advance (R1FH)** and **Velocity Timer (R19H)** registers are used to adjust the position of the commutator switching with regard to the phase torque characteristics of the motor. That is, it adjusts the 'switching angle' or 'ignition advance' of the stepper excitation.

When a stepper is stationary or switched at low speeds, maximum torque is obtained if the phases are switched at the crossover points of the static phase torque curves; i.e. when the encoder detects that the current step has completed, the next phase is excited. However, at higher speeds, the phase torque characteristics are distorted (due to the winding time constant and motional voltage). In these circumstances each change in excitation must occur earlier relative to the rotor position if maximum torque is to be achieved.

The theoretical optimum switching angle is given by:

$$\text{switching angle} = (\tan^{-1}\omega L/R)/p \qquad [11]$$

where

ω = angular frequency (rads/sec)
L = average winding inductance
R = total phase resistance
p = number of rotor teeth (usually 90/step angle)

Thus the optimum switching angle varies with the motor velocity. If a stepper motor is to be used at low speeds, the optimum switching angle (or phase advance) is zero, and both registers should be programmed to be zero. The registers should not be left un-programmed.

The switching angle should be calculated for the maximum velocity that the motor is to be driven. The phase advance in quadrature counts (or full counts if the controllers' status register bit 2 = 1) can then be calculated from the switching angle:

$$\text{Phase advance} = (\text{switching angle}/360) \times \text{No. of (quad) counts/rev} \qquad [12]$$

The Max. Phase Advance register can then be programmed with this value. The Velocity Timer register determines the 'slope' which the controller will use to calculate the phase advance used at intermediate speeds. The units for the Velocity Timer are the same as for the Max. Phase Advance register (i.e quadrature counts or full counts) but the value is calculated as follows:

$$\text{Advance} = N_f\ v\ \Delta t \qquad [13]$$

where

N_f = full encoder counts /revolution
v = velocity (revolutions/second)
$\Delta t = [16\ (R19H + 1)] / (2 \times 10^6)$ [14]

Example: a 200 step motor, L = 20.4mH, R = 7.4Ω, encoder = 500 counts/rev, is to be driven at 20Hz.

Optimum switching angle (at 20Hz) $= (\tan^{-1}(\omega L/R))/p$
$= [\tan^{-1}(2\pi{*}20{*}20.4{*}10^{-3}/7.4)]/50$
$= 19.1/50 = 0.38°$

Phase Advance = $(0.38° / 360) * 2000 = 2.1 \Rightarrow 2$ quadrature counts.

Thus the Max. Phase Advance register is programmed with '2' and the Velocity Timer register is calculated using equation 13 as follows:

$$2 = 500 \times 20 \times [16\ (R19H + 1)] / (2 \times 10^6)$$

which gives the R19H (Velocity Timer register) value as 24.

APPENDIX 3 - Synchronizing multiple AIM104-MOTION-1 boards

Each board has a SYNC line. A low going SYNC pulse resets the motion controller's 'Sample Timer Register'. At the beginning of each sample period the motion controller calculates the required output to the motor (depending on where the axis is and where it needs to go etc.). If no SYNC pulse is provided then the motion controller will run asynchronously.

The AIM104-MOTION-1 brings the SYNC line to pin 46 on the 50 way connector (PL3). Connecting together the SYNC lines of all the AIM104-MOTION-1 boards and pulsing the SYNC line synchronises the motion controller timers. The SYNC signal can be pulsed by writing to bit 2 of the configuration register of one of the boards (located at base address + 3). This ‘master’ board should have its internal/external link set to ‘internal’, all others to ‘external’. (see the ‘links’ section of this manual).

In order for multiple axes to follow a custom profile, the user will normally need to write a new position (or velocity) each sample time (to one or more of the axes). The new commands must not be written to the controllers during the first part of the sample period (see HCTL1100 data sheets) during which the controllers are calculating their new output values. The user can read the Sample Timer Register to avoid writing too early during a sample period. Since the Sample Timer Register counts down from its programmed value, the user can check if enough time has passed in the sample period to ensure the completion of the internal calculations.

The length of time needed by the HCTL-1100 to do its calculations is given by the Minimum Limits of Register 0FH (Sample Timer Register). See below. E.g. for Position Control Mode the user should wait for the Sample Timer to count down 07H from its programmed value.

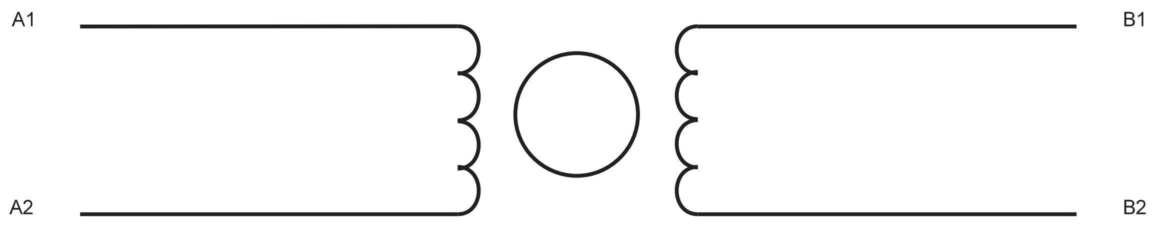
Control Mode	Reg. 0FH Contents Minimum Limit
Position Control	07H
Proportional Velocity Control	07H
Trapezoidal Profile Control	0FH
Integral Velocity Mode	0FH

APPENDIX 4 - Stepper Motor Configurations

There are two commonly used types of stepper motor, the Bipolar stepper motor and the Unipolar stepper motor, both of which may be used with the AIM104-MOTION-1 board.

BIPOLAR STEPPER MOTORS

Wiring of bipolar stepper motors to the AIM104-MOTION-1 board is simple. Connect the four wires to A1, A2, B1 and B2 as follows.

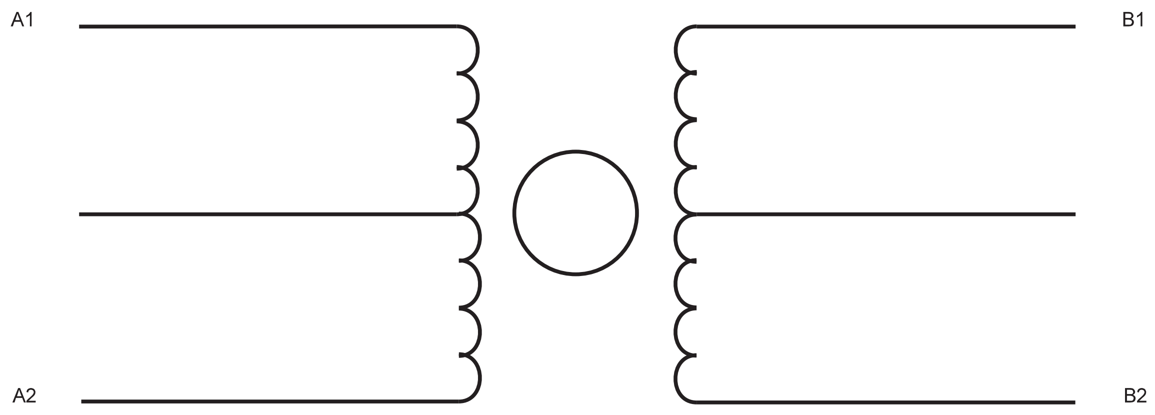


UNIPOLAR STEPPER MOTORS

Wiring of unipolar stepper motors to the AIM104-MOTION-1 board depends upon whether it is to be wired in *series* or in *parallel*. Six wire unipolar stepper motors can only be wired in series. 8 wire unipolar stepper motors can be connected in either series or parallel.

WINDINGS IN SERIES

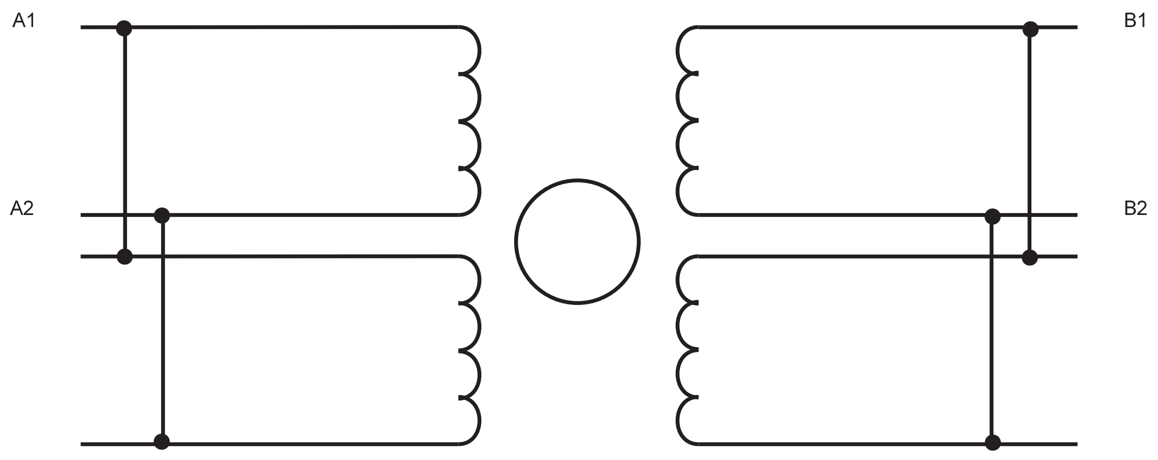
Connect A1, A2, B1 and B2 to the motor as follows:



Configuring a unipolar stepper for bipolar operation gives superior torque in comparison to traditional unipolar drive methods because all four windings can be active at the same time.

WINDINGS IN PARALLEL

Connect A1, A2, B1 and B2 to the motor as follows:



The parallel configuration can give superior motor torque than the series winding because the winding current may be increased by a factor of √2 because there is double the cross-sectional area of winding wire.

Appendix 5 - Digital Filter Parameters

The choice of values for the digital filter parameters (A, B and K) are of critical importance to the stability and dynamic response of the control loop. The best values will depend mainly upon the characteristics of the motor used and on the choice of control mode (e.g position control mode, integral velocity mode).

DC Servo Motors

The table below shows how changing the filter parameters, changes the systems response.

Increase in parameter	Stability	Response time	Stiffness (1/dead band)
A	Better	Faster	Decreases
B	Slightly better	Faster	Decrease
K	Worse	Faster	Increases
t (sample period)	Worse	Slower	Decreases

Thus for good stability, the sample period should generally be as small as possible.

Note: A good approximation of the loop response can be obtained by a spreadsheet model of the system using the loop components, e.g. Digital compensation filter equation (appendix 2), Motor Voltage constant (Kmv) and Mechanical Time constant. Theses values can be used to calculate the output of the controller and the subsequent movement of the motor.

Stepper Motors

The filter parameters do not have the same effect on the performance of stepper motors. The most important parameter in a stepper system is the Gain, K. The gain must be big enough so that the driver output is large enough to move the motor one step (or half-step) when the positional error is small.

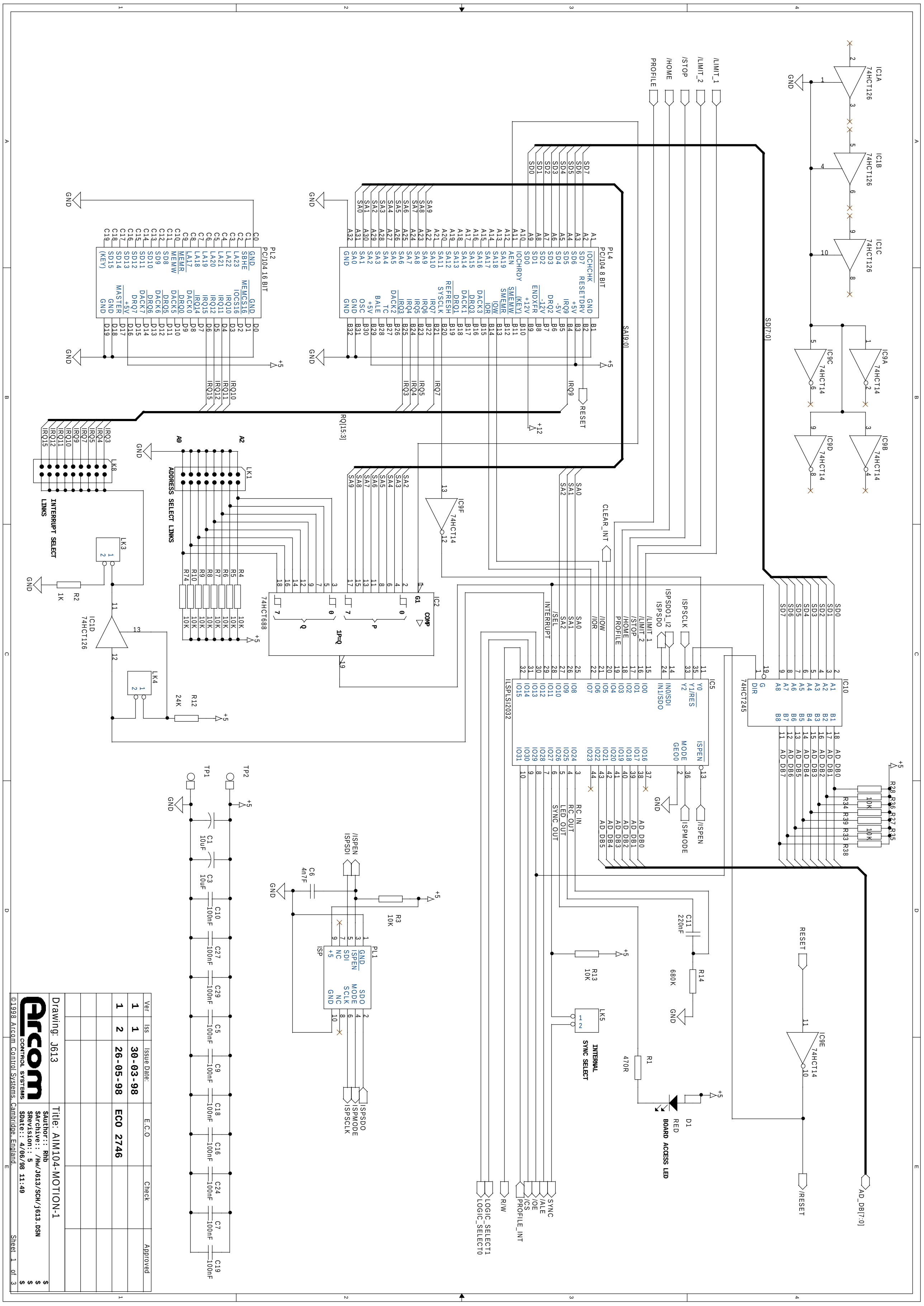
However, when a stepper motor moves a whole step (or half-step), there is some spring on the movement. That is the rotor will go past the detent position before springing back. The greater the drive throught the motor coils, the larger the overshoot.

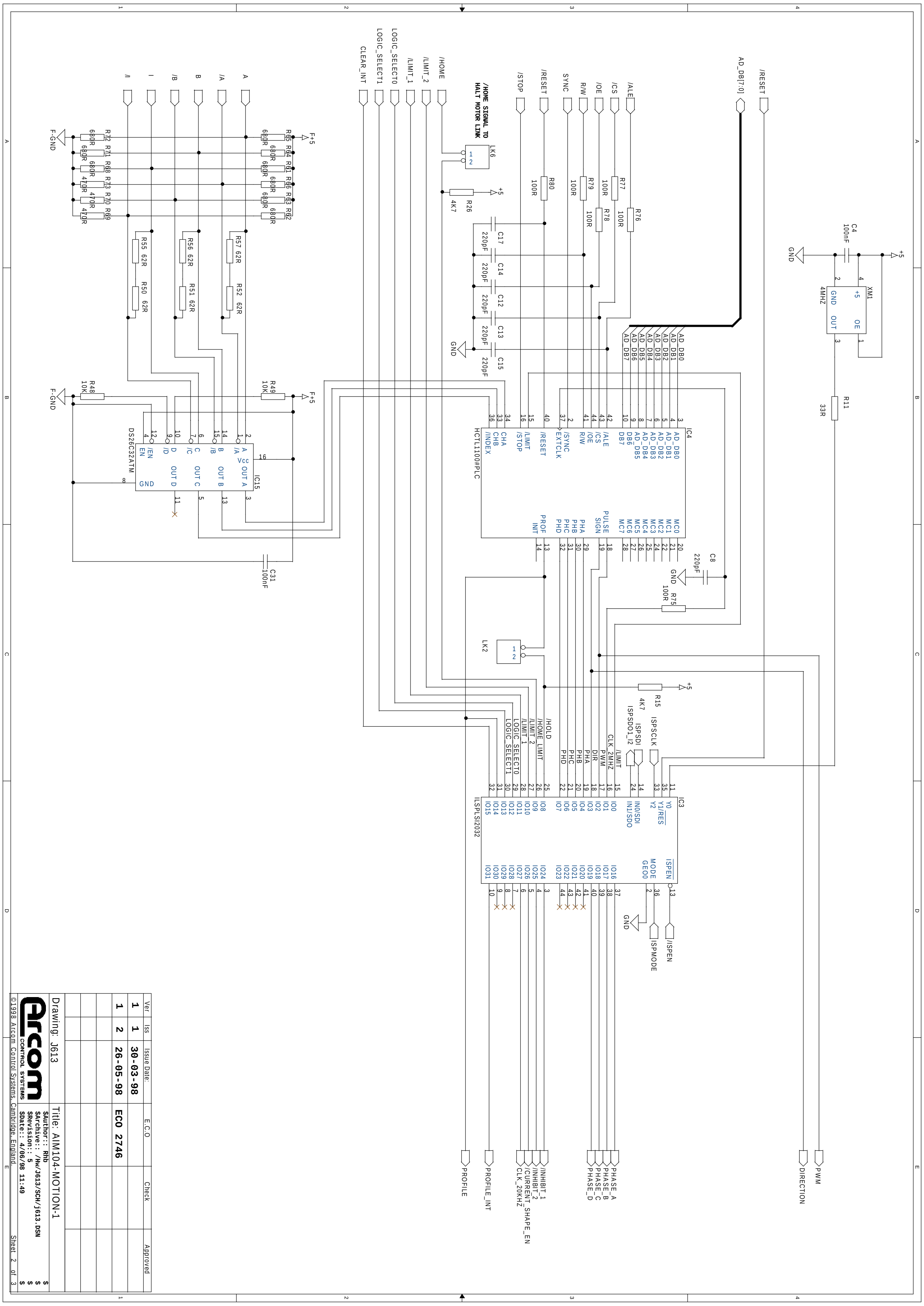
Ideally, when there is a position error of even one half-step, the AIM104-MOTION-1 output will be large enough to drive the rotor to the next detent. As soon as the encoder reaches the command position, the positional error will be zero, the output will become zero and the motor will stop.

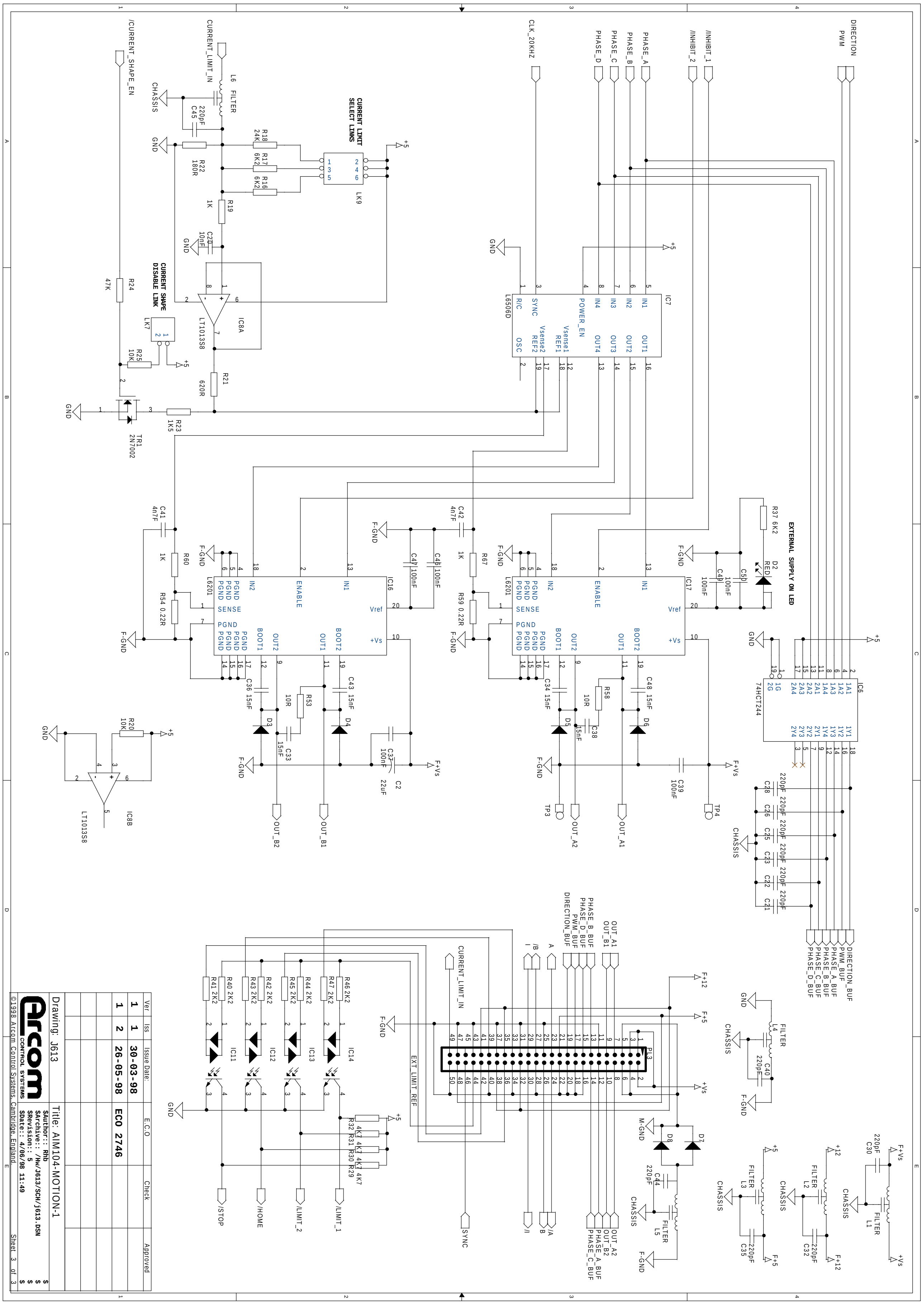
However, if when the position error is only one step (or half-step) the driver output is too large, then when moving to the last step, the rotor may overshoot enough so that the encoder feedback causes the controller to drive the motor back one step (or half-step). This situation would result in the stepper motor oscillating between two step (or half-step) positions constantly when the controller is trying to hold the motor in position.

This problem could be solved by reducing the gain K or reducing the external voltage supplied to the driver circuits, to reduce the overshoot. Alternatively, the system damping could be increased. This problem is more pronounced in poorly damped systems, such as an unloaded motor.

Circuit Diagrams




IC4
HCTL1100#PLC
IC15
DS26C32ATM
IC3
ILSPLSI2032
XM1
4MHZ
LK6
/HOME SIGNAL TO HALT MOTOR LINK
LK2
AD_DB[7:0]
/RESET
/STOP
SYNC
R/W
/OE
/CS
/ALE
/HOME
/LIMIT_2
/LIMIT_1
LOGIC_SELECT0
LOGIC_SELECT1
CLEAR_INT
A
/A
B
/B
I
/I
F+5
F-GND
GND
+5
PROFILE
PROFILE_INT
/INHIBIT_1
/INHIBIT_2
/CURRENT_SHAPE_EN
CLK_20KHZ
PHASE_A
PHASE_B
PHASE_C
PHASE_D
DIRECTION
PWM
ISPSCLK
ISPSDI
ISPSDO1_I2
/ISPEN
ISPMODE
Drawing: J613
Title: AIM104-MOTION-1
$Author:: Rhb
$Archive:: /Hw/J613/SCH/j613.DSN
$Revision:: 5
$Date:: 4/06/98 11:49
Ver Iss Issue Date E.C.O Check Approved
1 1 30-03-98
1 2 26-05-98 ECO 2746
©1998 Arcom Control Systems, Cambridge, England
Sheet 2 of 3
arcom CONTROL SYSTEMS


Drawing: J613
Title: AIM104-MOTION-1
$Author: Rhb $
$Archive: /hw/J613/SCH/j613.DSN $
$Revision: 5 $
$Date: 4/06/98 11:49 $
Ver | Iss | Issue Date | E.C.O | Check | Approved
1 | 1 | 30-03-98
1 | 2 | 26-05-98 | ECO 2746
Sheet 3 of 3
© 1998 Arcom Control Systems, Cambridge England
CURRENT LIMIT SELECT LINKS
CURRENT SHAPE DISABLE LINK
EXTERNAL SUPPLY ON LED
L6506D
L6201
74HCT244
LT1013S8
2N7002