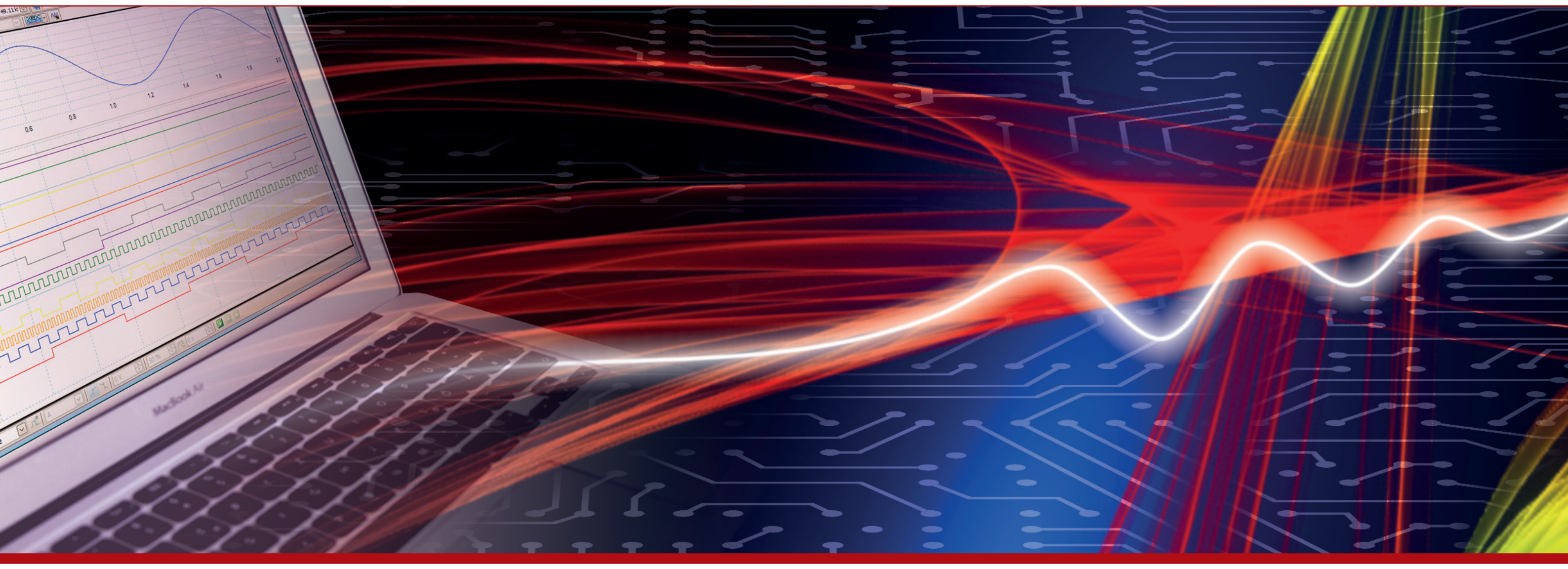
Product Datasheet - Technical Specifications



More information in our Web-Shop at ▸ **www.meilhaus.com** and in our download section.

Your contact

Technical and commercial sales, price information, quotations, demo/test equipment, consulting:

Tel.:	**+49 - 81 41 - 52 71-0**
FAX:	**+49 - 81 41 - 52 71-129**
E-Mail:	**sales@meilhaus.com**

Downloads:
www.meilhaus.com/en/infos/download.htm

Meilhaus Electronic GmbH
Am Sonnenlicht 2
82239 Alling/Germany

Tel.	**+49 - 81 41 - 52 71-0**
Fax	**+49 - 81 41 - 52 71-129**
E-Mail	**sales@meilhaus.com**

www.meilhaus.de

USER'S MANUAL
INTELLIGENT MOTION CONTROLLERS

FOR PCI BUS

MAXk

OMS Motion, Inc.
15201 NW GREENBRIER PARKWAY
B-1 RIDGEVIEW
BEAVERTON, OR 97006
PHONE 503-629-8081
FAX 503-629-0688
mailto:sales@omsmotion.com
WEB SITE http://www.omsmotion.com/

3301-1900000

Rev. B

TABLE OF CONTENTS

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The OMS Motion, Inc. MAXk family of motion controllers are high performance PCI bus-based products and are in compliance with the “standard length” universal PCI Bus Specification (Revision 2.2). The MAXk motion controller can manage up to 10 axes of open loop stepper, closed loop stepper or servo systems, in any combination. The OMS Motion, Inc. MAXk controller synchronizes all independent or coordinated motion of up to 10 axes, while incorporating other critical signals, such as hard or soft limits, home, and other digital and/or analog I/O signals, to provide the motion solutions to perform virtually any task. With high level functionality, such as circular and linear interpolation, multi-tasking, custom profiling, etc., the MAXk can satisfy the needs of most any motion control application. See Appendix C “Ordering Information” for specific MAXk family models.

The MAXk communicates as a “slave only” device and functions as a motion co-processor to the PCI host. It utilizes patented, proprietary technology to control the trajectory profile, acceleration, velocity, deceleration and direction of selected axes. In response to commands from the host computer, the MAXk controller will calculate the optimum velocity profile to reach the desired destination in the minimum time, while conforming to the programmed acceleration and velocity parameters. In addition, the MAXk can provide motion control information such as axis and encoder position, as well as the state of over travel limits, home switch inputs, and done interrupt flags. The MAXk motion controllers utilize a Power PC processor, configured to operate as an efficient and powerful co-processor with the PC host via the PCI Bus at 33 MHz.

The stepper control of the MAXk produces a 50% duty cycle square wave step pulse at velocities of 0 to 4,000,000 pulses per second and an acceleration of 0 to 8,000,000 pulses per second per second. The servo control utilizes a 16-bit DAC and outputs either +/- 10V or 0 to +10V. The encoder feedback control can be used as feedback for the servo PID, position maintenance for the stepper axes or as strictly a position feedback of any axis. The encoder input supports differential or single ended quadrature TTL signals at a rate of up to 16 MHz. The MAXk motion controller has 4 general purpose analog inputs that utilize a 16-bit ADC, with a DC range of –10 to + 10 VDC. Complete specifications for MAXk can be found in Appendix C.

The MAXk command set employs two or three ASCII character commands which can be combined into character strings. Using virtually any programming language, these ASCII command strings can be sent to the MAXk Motion Controller over the PCI bus.

1.2. SYSTEM OVERVIEW

The MAXk is a standard length PCI module (12.83” x 4.20” x 0.475”). The communication interface is accessed through the PCI bus and is compliant with the PCI Bus Specifications, Revision 2.2, see Figure 2-1. The MAXk receives power (3.3V, 5V, +/-12VDC) from the host computer by the PCI bus.

The MAXk utilizes an optimally configured Power PC RISC based 32-bit micro-controller and FPGA technology for extensive logic integration and flexibility. The firmware, which resides in flash memory, can be upgraded through the communication interface without having to remove the controller from the system. 32MB of system RAM is used for firmware and data storage.

There is a jumper block J5 on the MAXk that allows for setting the board number. All general purpose digital and analog I/O signals and all motor control signals are available on the two 100-pin connectors (J1, J2). Each digital I/O bit can be set as an input or output and is controlled by firmware commands, so there are no jumper to set.

Data communication is performed by sending and receiving strings of data (ASCII characters) through the DLL or driver. The device driver provides handshaking information for writing to the shared memory registers. Requests by and responses to the driver handle some status information like; error conditions, motion complete, and so on. See also Fig. 3.1 Functional Communications Flow.

The MAXk bus interface uses PCI memory technology to provide a fast communication channel for the commands from the Host PC as well as feedback of motion parameters, such as encoder positions. Commands may be written to this RAM by the host, thus eliminating the bottlenecks of I/O and port based communications. Critical motion parameters such as position and velocity are available, allowing the host to interrogate these parameters in real time while the motion is in progress. All of the data can be captured within the same update cycle.

Interrupt control and other data are available through blocks of dedicated registers. These registers report status on controller flags over travel limit, done flag, and encoder slip for each axis. The device driver processes and interrogates these registers then initiates the appropriate action.

Some commands may be passed to the MAXk while bypassing the communications channel. These direct commands cause an immediate interrupt and may be used for critical commands such as an abort. Each axis may perform individual unrelated moves or the controller can be coordinated as required by the application.

DLLs are provided to allow applications written in high level languages to communicate with the controller. Software provided by OMS Motion, Inc. directly supports the use of Microsoft C, C++ or Visual Basic. In addition, any language that has a mechanism for utilizing a standard Microsoft DLL Library can be used for application development.

The MAXk I/O Breakout Modules, the IOMAXnet, provide an efficient means of connecting the MAXk signals to external devices.

More details on the functionality of the controller are included in the following chapters.

2. GETTING STARTED

2.1. INSTALLATION

For installation of MAXk you will need a computer with a PCI bus that is compliant to the PCI Bus Specification, Revision 2.2. The MAXk is a universal PCI board and can be installed in either a 5VDC PCI slot or a 3.3VDC PCI slot.

Read through the following two sections before beginning the installation. Do not turn on the power to the PC until you have properly configured the controller per the following instructions. Note that the header J5 is used to choose the controller number. The MAXk is set for controller number 1 from the factory.

Though the MAXk is a low power device, there should be ventilation, including forced air, around the circuit board. The MAXk will draw all of its power from the PCI bus, so no external power supply is needed.

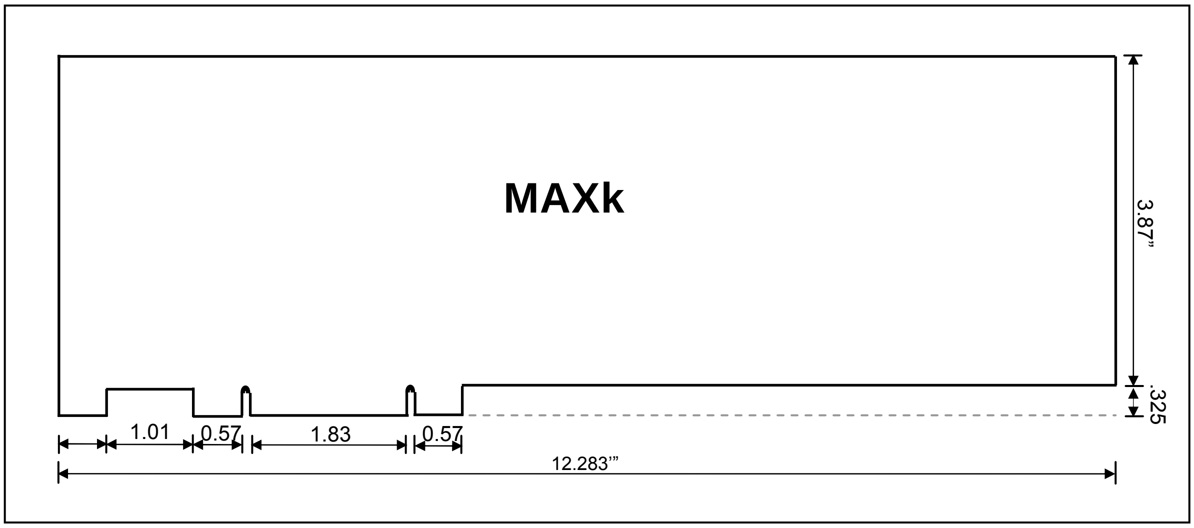


Figure 2-1 Board dimensions

2.2. TO PREPARE FOR INSTALLATION

Note: The Board Number Select Jumper J5 must be set BEFORE power up. (See Figure 2-3 for MAXk component locations.)

CAUTION:

The MAXk is a static sensitive device and standard Electro Static Discharge (ESD) techniques are required when handling and installing the MAXk.

2.3. BASIC CARD CONFIGURATION

If the MAXk is to be installed with other MAXk boards, the Board Number Select Jumper (J5) must be set to a different value on each board. The Board Number Select Jumper block is located near the bottom center of the board and is labeled J5 (See Figure 2-2). Set the first MAXk board identification number to 1, the second MAXk to 2, and so forth. The default value is all jumpers off, which identifies the board as controller #1. The jumper block represents the board number in binary form (4-bit). See Figure 2-2 - BOARD SELECT Jumper (J5)

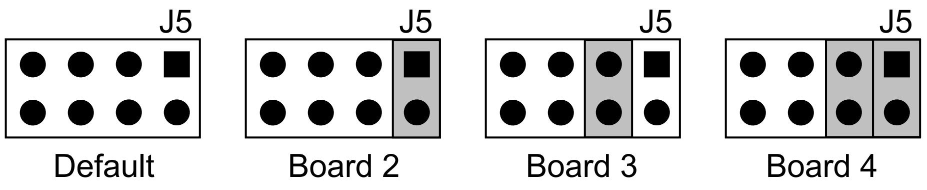


Figure 2-2 - BOARD SELECT Jumper (J5)

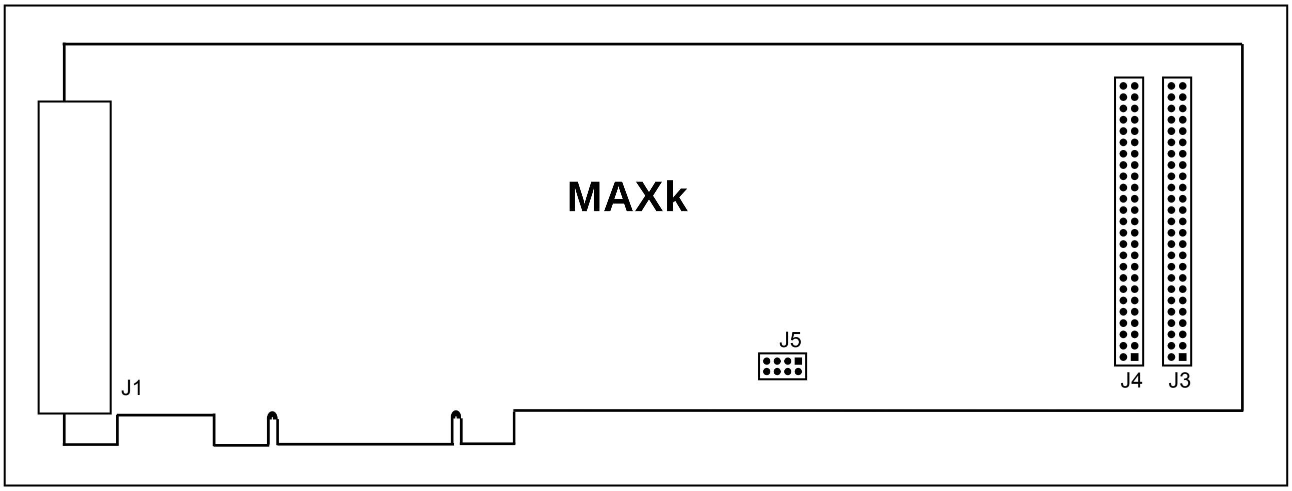


Figure 2-3 MAXk connector and jumper locations

2.4. CONFIGURING THE CARD FOR USE WITH ENCODERS

Quadrature encoder with TTL level outputs can be connected directly to the appropriate axis (on IOMAXnet connected to J1 and/or J2 connector resp.). The MAXk has built-in biasing to allow single ended encoders for each axis on board. Single ended operation may be limited by cable length and encoder velocity

2.5. HARDWARE INSTALLATION

- Turn off power to the PCI computer and disconnect its power cord from the wall socket.
- (Note: Some PC's have power on the board even if the power switch is turned off.)
- Remove the cover of the computer to gain access to the PCI bus.

Caution

ESD Warning: The MAXk, as well as most computers, are sensitive to Electro Static Discharge (ESD) and may be damaged if proper precautions are not taken to avoid ESD. Use properly grounded ESD mats, wrist-straps and other ESD techniques to prevent damage to the controller and/or computer.

- On the MAXk board, configure board number selection jumper (J5) as needed.
- For controllers with more than 5 axes, attach the ribbon cable to the J3 and J4 (Figure 2-3) connector on the board. Make sure to align pin-1 correctly at the connector on MAXk.
- Align the MAXk with the PCI slot of the computer and insert the MAXk fully into the slot. (Remember, the MAXk will require a full length slot.)
- Double check the board to ensure it is properly seated in the connector.
- Use a screw to fix the bracket on the MAXk to the computer's chassis.
- For controllers with more than 5 axes:
- Position the J2-Bracket with the connector so that it can be accessed from outside the PC.
- Screw down the J2-Bracket to the computer's chassis
- Double-check that the MAXk is correctly installed and that the ribbon cables are correct.
- Once you are sure everything is installed and configured correctly, replace the cover to the PC.
- Replace the power cord and turn on the computer.
- Allow the computer to boot up.

Establish communication with the controller board **before** wiring external components to the board (i.e. drivers and motors). This can be done by using the MAXkComm.exe utility.

DO NOT make wiring connections to the controller board with power applied to the board.

Caution

2.6. SOFTWARE INSTALLATION

OMS provides drivers for Windows NT, 2k, XP, Vista 32-bit and Win7 32-bit for other operating systems please contact OMS Motion, Inc, refer to Appendix B.

For Windows NT, XP, 2k, Vista & Win7

After installing the MAXk in the chassis, apply power to the host PC and insert the software support disk or CD-ROM supplied by OMS or download the software from the OMS website (http://www.omsmotion.com/). Follow the installation instructions found in README.TXT or README.DOC. The instructions will show you how to properly install the device driver and appropriate DLL.

To begin communicating with the MAXk, run the OMSSuite.EXE utility and select the MAXk controller number you selected with the Board Number Select jumpers. You can begin interactively sending commands and receiving responses immediately if all has been properly installed.

For backward compatibility with MAXp, installed MAXp drivers work properly with MAXk although without the added features of the MAXk.

Type WY and observe the response from the MAXk. If you are communicating to the MAXk it would return its version number, number of axes, FPGA version number, etc. You should receive a reply similar to “MAXk-8000, Ver: x.xx, S/N: 000001, FPGA:20” from the MAXk. If you receive nothing, double check that the MAXk is fully seated in the chassis and the device drivers are installed properly. For technical support, refer to Appendix B for contact information.

2.7. CONNECT TO STEPPER MOTOR SYSTEM

The MAXk control signals are located on the J1 connector and J2 connector where applicable. This section will explain how to connect a stepper motor driver to the controller board.

Begin this procedure with a MAXk controller board installed in your system. Be sure that communication to the board has been established. This can be checked by issuing a WY command to the board and verifying that the board responds with its model type and revision levels (i.e. MAXk-4000 ver 1.00 S/N 0001).

NOTE: Reference section 2.6 SOFTWARE INSTALLATION

Once communication has been established with the controller, shut down the system and turn power off to the controller board.

NOTE: It is not recommended to continue with the hardware connection if communication has not been established.

Connect the motor phase signals from the motor to the stepper driver output signals. Use the motor and stepper driver manufacturer's manuals for instructions.

Now, connect the controller signals from J1 of the MAXk, or from IOMAXnet, if it is used, to the stepper driver. Short cable lengths and shielded cables are recommended for improved signal integrity and reduction in signal noise.

NOTE: Using the IOMAXnet interface module is strongly recommended as it provides an easy way to connect to the 100-pin connectors J1 and J2 on the MAXk.

Use a shielded 100-pin cable to connect the IOMAXnet to the MAXk. From the terminal block on the IOMAXnet connect the appropriate wires to your motor drivers and system I/O.

Attach the STEP outputs from the controller to the STEP inputs on the stepper driver. Do the same for DIR signals.

Next, connect an external power supply (which is OFF) to the stepper driver. Again, refer to the manufacturer's manual for instructions. (Note that power supply requirements differ from driver to driver.)

Once all wire connections have been made, power can be restored to your system. It is recommended that you bring the controller board up first (so it is in a known state), and then apply power to the stepper driver.

Refer to Figure 2.5 for an example wiring diagram of OMS' MAXk connected to the OMS PMD4-m stepper driver on the X axis.

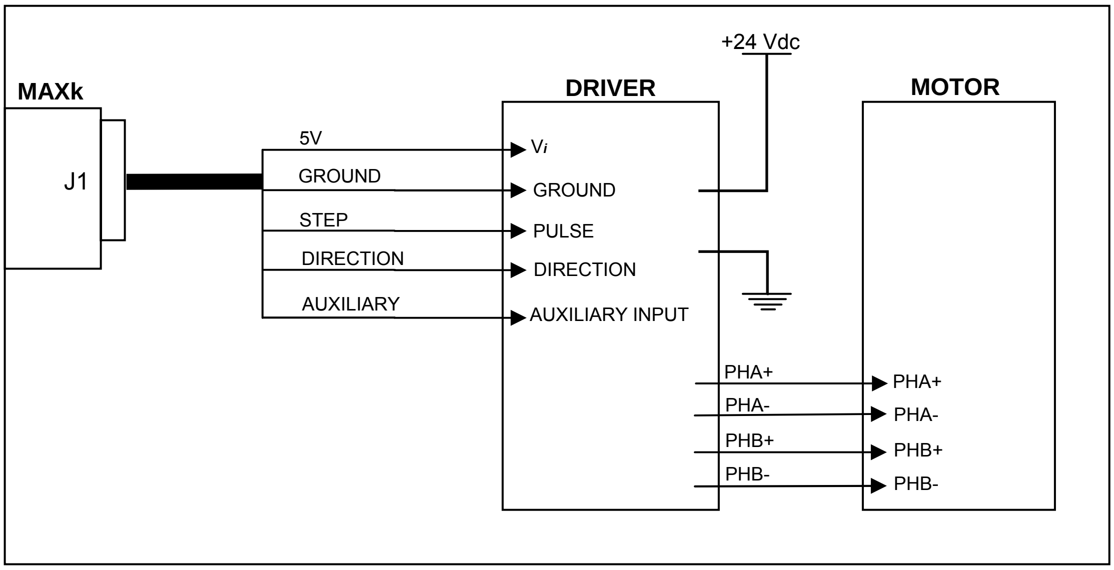


Figure 2-4 Example Wiring Diagram of MAXk Controller

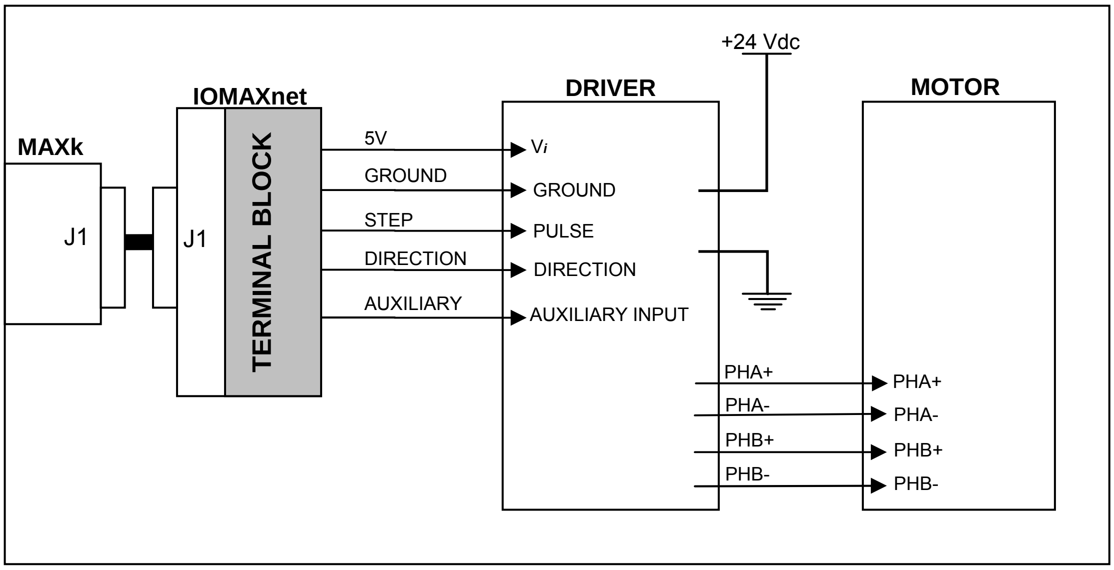


Figure 2-5 Example Wiring Diagram of MAXk Controller via the IOMAXNET Interface Module

2.8. CONNECT AND CHECKOUT THE SERVO SYSTEM

Servo systems tend not to respond gracefully to connection errors. You can reduce the chance of making connection errors by following a step-by-step procedure:

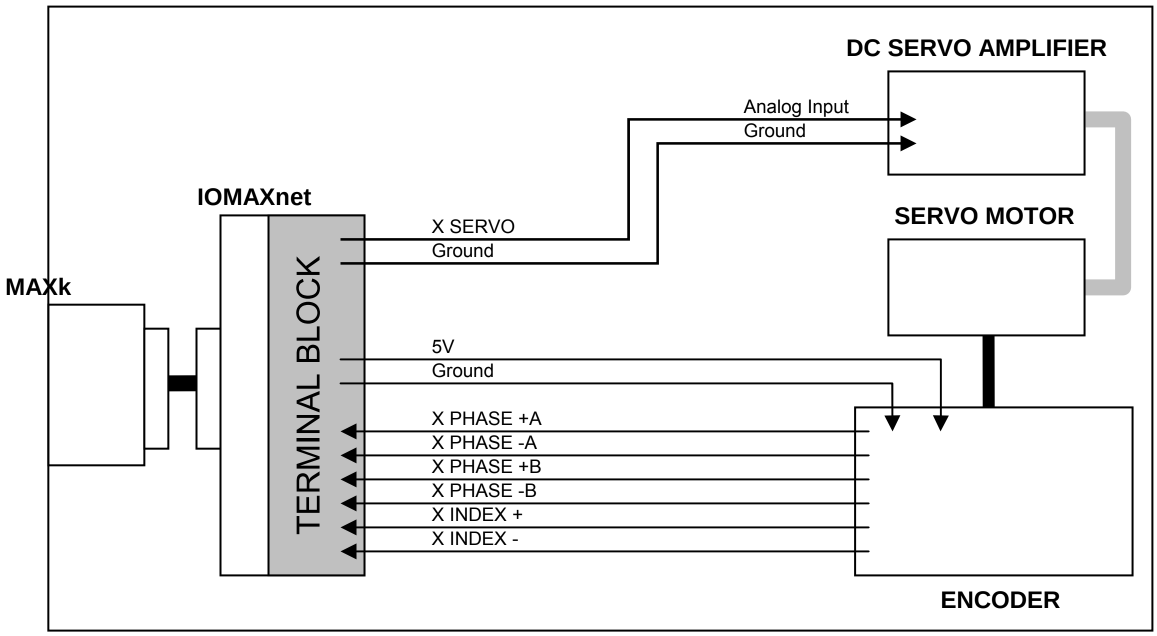


Figure 2-6 EXAMPLE OF WIRING DIAGRAM OF MAXK CONTROLLER VIA THE IOMAXNET INTERFACE MODULE TO SERVO MOTOR

Caution

The servo motor may jump or spin at a very high velocity during connection and configuration. The motor should be restrained by some means before beginning this procedure. Keep hands and clothing clear of the motor and any mechanical assemblies while performing this procedure.

It is recommended that the motor shaft not be connected to the physical system until you are sure you have control over the motor.

2.9. CONNECT AND CONFIGURE THE MOTOR/AMPLIFIER

1. Connect and configure your amplifier per the manufacturer's instructions for "Torque" or "Open-Loop" mode.
2. With the motor and amplifier power turned off, connect the MAXk to the amplifier.
3. Balance your motor:
    a. Configure the axis as a servo axis by sending the "PSM" command.
    b. Using a voltage meter, verify that the command signal from the MAXk is less than 500mV. If it is not, send the command "KO0;" to the MAXk and recheck the voltage. If the voltage is still too high, contact OMS Motion, Inc.' Technical Support department for guidance.
    c. Turn on power to the amplifier and then to the motor.
    d. Adjust the balance setting of your amplifier (if equipped) until the motor stops moving.
    e. If the motor continues to revolve or your amplifier has no balance adjustment:
        i. Send the command "KO100;" to the MAXk.
        ii. If the motor spins faster, reduce the command parameter and resend the command, e.g. "KO50;"
        iii. If the motor spins slower but does not stop, increase the command parameter and resend the command, e.g. "KO150;"
        iv. Continue adjusting and resending the KO command until the motor comes to rest. Write down the final KO value for later reference as your "zero" setting.
4. Maximize your system's usage of the MAXk's DAC (this method works only with incremental encoders, skip it if you use absolute encoder only on that axis):
    a. Connect the servo encoder to the MAXk. (See section 4.4 on incremental encoder feedback)
    b. Set the signal/command gain of your amplifier to its minimum setting.
    c. Send the "KO3277;" command to the MAXk and observe the velocity of the motor. The output of MAXk will be near 1VDC.
    d. If the motor does not move at all, your amplifier does not work well at a low velocity. In this case, adjust the signal/command gain of the amplifier to approximately 20% of maximum or until the motor begins to move.
    e. Using a frequency meter, measure the pulse rate of Phase A of the encoder. The frequency measured is ¼ of the actual pulse rate.
    f. Adjust the signal/command gain of the amplifier until the pulse rate of Phase A is approximately 10% of your desired peak operational velocity. If the pulse rate is already greater than 10% of peak, your amplifier is not designed for low velocity motion and you will likely have some difficulty tuning your motors.
    g. Send the "KO-3277;" command to the MAXk and recheck the velocity. You may need to readjust your amplifier. If so, do not reduce the signal/command gain – only increase the setting as needed. Increasing the gain will not impair the forward peak velocity but reduction will.
    h. Send the KO command with the "zero" value to the MAXk.

5. Verify the direction of your servo encoder:
    a. Send the "LP0; KO2000;" command to the MAXk.
    b. Send the "RE;" command to the MAXk and observe the response.
    c. If the response is positive, no further action need be taken; go to step 6.
    d. If the response is negative, your encoder or analog output must be reversed use one of the methods below.
        i. Use EDI/EDN to invert/normalize encoder direction or
        ii. Use SVP-/SVP+ to invert/normalize PID analog output (inverts values of KO and KOD) or
        iii. if your incremental encoder produces a differential signal, swap Phase B+ with Phase B- and repeat from step (a.) above.
        iv. If your incremental encoder produces a single-ended (or TTL) signal, swap Phase A with Phase B and repeat from step (a.) above.
    e. If the RE response is still negative, contact OMS Technical Support for assistance.
6. Repeat from step 1 for the other servo axes.
7. Remember to set KO for each axis at every power-up unless you store the values in Flash.

NOTE: Most encoder problems are caused by lack of power or incorrect connections. If the encoder position changes by only 1 count, this is an indication that one of the phases is not connected.

Do not proceed until you perform all the steps in this procedure, ensure that the outputs of the MAXk are as described, and ensure that the encoder is operating correctly.

2.10. TUNE THE SYSTEM

2.11. INTRODUCTION

The following is an introduction to tuning a servo motor and the basics of the process of doing it. Tuning a servo system is the process of balancing three primary gain values Proportional, Integral, and Derivative in order to achieve optimum system performance.

In a closed loop system, an error signal is derived from the command position and actual position, amplified, and then supplied to the motor to correct any error. If a system is to compensate for infinitely small errors, the gain of the amplifier needs to be infinite. Real world amplifiers do not possess infinite gain; therefore, there is some minimal error which cannot be corrected.

The three primary gain values used in servo systems are **P** (proportional), **I** (integral) and **D** (derivative). The "P" term is used as a straight gain factor to get the system response "in the ballpark." The "I" term defines how quickly the system will respond to change. The "D" term is a dampening term. This term defines how quickly the system settles at its desired position without oscillating.

The effects of these parameters can be seen when looking at the system's response to a step change at the input. The shape of the step response falls into one of three categories: under damped, critically damped or over damped. Over damped systems are slow to reach their final value and produce little or no oscillation. Critically damped systems reach final value quickly, without overshoot. Under damped systems reach final value quickly, but have various degrees of "ringing" or oscillation, that decay to zero over time. Ideally, a system should be critically damped, allowing for the fastest response time with the least amount of oscillation.

2.12. TUNING ASSISTANT

OMS' Tuning Assistant utility is provided to assist the user in finding the right combination of parameters. This utility plots the motor's response. The user can analyze this data to arrive at the right servo parameters for their servo system.

2.13. MANUAL TUNING

In most motion control applications the optimum tuning of the servo system is achieved through a manual tuning process. Auto-tuning algorithms typically can only get the system parameters close and require manual steps to fine tune the parameters. An empirical trial and error approach will be discussed first.

There are some system parameters that need to be determined before attempting to tune a motor. The encoder resolution (counts per revolution) is one element to be determined. Another is the system's maximum velocity. Note that a motor should never exceed 90% of the motor's maximum rate rpm. If the system requirement is for a velocity higher than 90% of the motors top rpm, then another motor with higher rpm capability should be used.

The system's maximum acceleration is determined several different ways. The best method is to determine the system time constant, which includes "hitting" or "bumping" the motor under system load and measuring the time from 0 rpm to maximum rpm and divide this value by 5. The maximum acceleration is either 2.5 times this value or is based on the system requirements for handling the load as defined in the operating specifications of the system. This value is always lower than the calculated value and if this acceleration value is not high enough then a different motor/amplifier with more power or bandwidth should be utilized.

The MAXk can control either current mode or voltage mode amplifiers. The #UR command sets the servo update rate of the MAXk to one of the following rates: 976.6µs, 488.3µs, 244.1µs, 122.1µs. This affects the responsiveness of the system. High "Following Error" can be compensated for using the feedforward coefficients (KV and KA commands) explained later in this section. There are some general formulas that have been developed to determine acceptable "Following Error" for both current and velocity mode systems:

Current mode:

KP "Following Error" = (3°/360°) × (counts per revolution)

Voltage mode:

KP "Following Error" = (90°/360°) × (counts per revolution)

It is obvious that the voltage mode allows for much greater "Following Errors" than the current mode. This value is the "Following Error" when the motor is at peak velocity and will be used when determining the proportional gain (KP).

The "Following Error" for the integral term (KI) or long-term gain value will follow the guidelines below:

Current Mode:

KI "Following Error" = 0 counts

Voltage Mode:

KI "Following Error" = (80°/360°) × (counts per revolution)

While still in open-loop mode (CL0;) use the KO command to zero the motor. This variable is used to provide a constant output that will compensate for torque offset from the load. So, when the system should be stationary, the necessary voltage will be sent to the amplifier to cause the motor to maintain position. With the correct KO value, the motor should successfully maintain a zero position.

KO is the offset coefficient used while in closed-loop or open loop mode, hold on (HN). You should have determined the correct value the KO variable before beginning to tune the PID filter.

The values for KO range from –32767.00 to 32767.00.

Set the previously determined values for maximum velocity, maximum acceleration and the move distance for a trapezoidal profile with at least a 20% flat spot at peak velocity. Use the following formula to determine the move distance:

Profile distance = ((peak velocity)^2/(2×acceleration))×2.4

Example: ((50,000)^2/(2×500,000))×2.4 = 6,000

Set the KD and KI variables to 0, and the KP variable to 1, and execute the move by sending the move commands to the MAXk.

Example:
```
MR6000;
GO;
```

Adjust the KP term while repeating the above move command until the "Following Error" at the flat spot of the profile is acceptable. If the motor becomes unstable prior to obtaining the optimum KP term, then increase the KD term until the motor stabilizes.

Example:
```
LP0;
KP1;
CL1;
MR6000;
GO;
LP0;
KP2;
CL1;
MR6000;
GO;
LP0;
KP4;
HN;
MR6000;
GO;
LP0;
KD10;
CL1;
LP0;
KP8;
CL1;
MR6000;
GO;
LP0;
KD100;
CL1;
```

The values in the above example are totally arbitrary and may vary drastically with different systems. The LP0 command is used to set the position error to 0.

The values for KP range from 0.00 to 32767.00.

Once the KP term has been obtained, it can be used to determine the initial value for the KI term. Set the KI and KU variables to 4 times the KP value. The KI term is a gain applied to the accumulated position error over time. The KU variable limits the amount the KI term can contribute to the PID. Continue executing the motion profile and raising the KU term until the long-term "Following Error" is acceptable. This error can be measured at the two knees of the motion profile. Increasing the KI term will increase the response time of your system. The motion profile should also have a steeper slope as KI increases (see figures 2-9 and 2-10 below). However, as KI increases the system can also become unstable. When the increased KI values cause unacceptable instability, increase the KD parameter. This will increase the dampening on the system's motion profile; therefore, reducing oscillation or "ringing". Continue adjusting the KI/KU and KD terms until the proper response time is obtained.

The values for KI range from 0.00 to 32767.00.

The values for KU range from 0.00 to 32767.00.

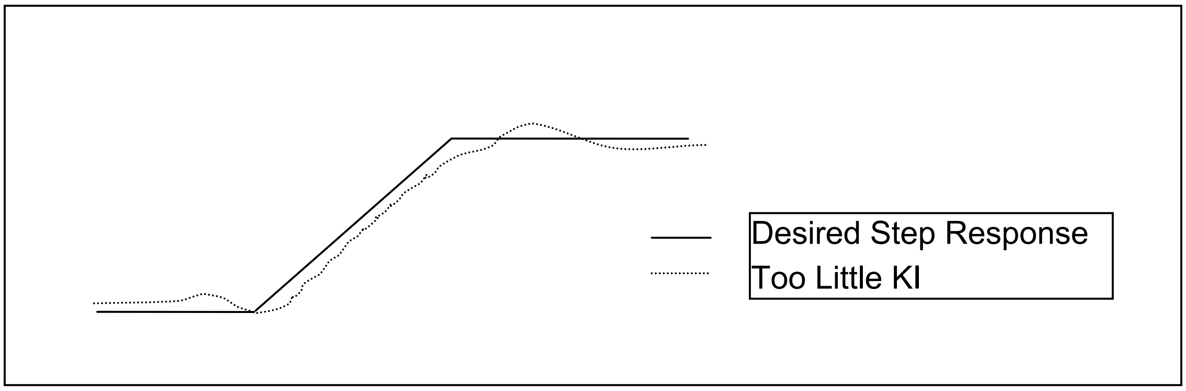


FIGURE 2-7

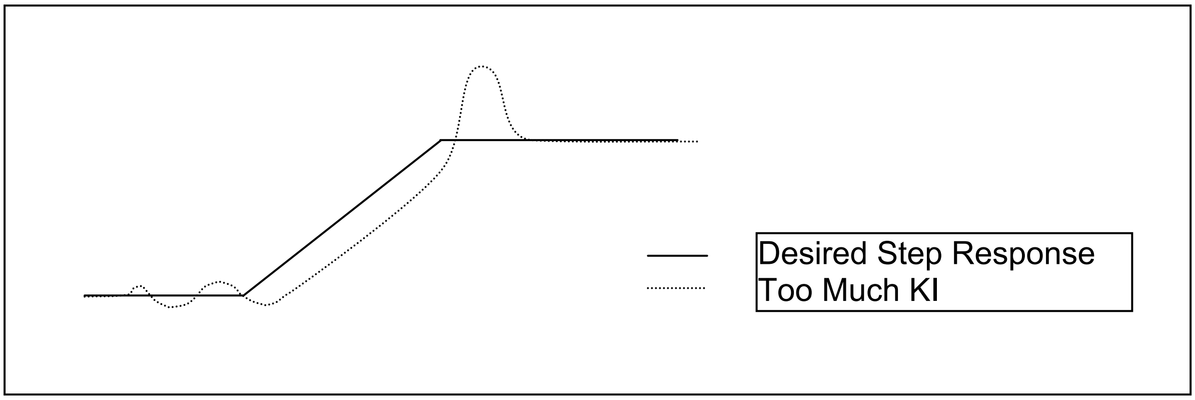


FIGURE 2-8

If you are getting too much "ringing" in the motion profile, then increase KD to help dampen the system's response. If, instead, the system is over-damped and is reaching the final velocity too slowly, then reduce the KD parameter. Optimally, the system's motion profile should show the motor reaching the desired velocity as quickly as possible without overshoot and oscillation ("ringing").

The values for KD range from 0.00 to 32767.00.

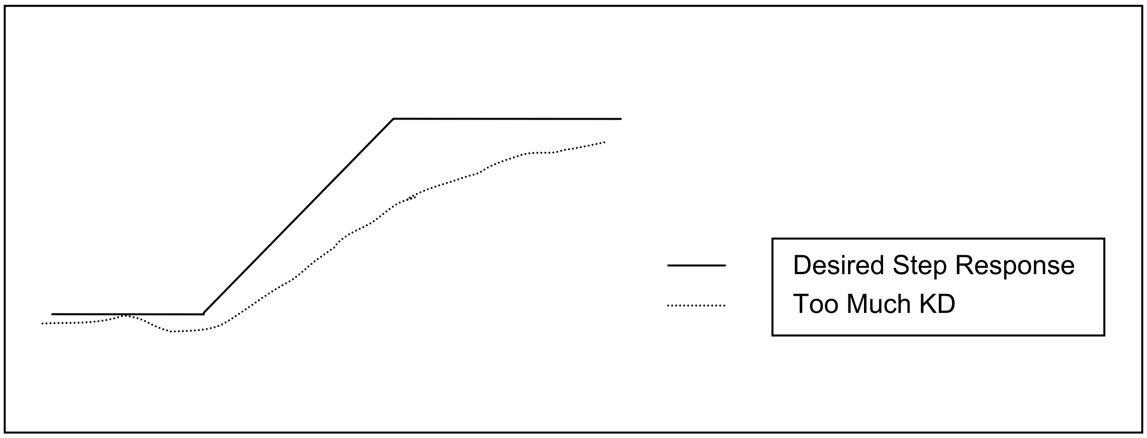


FIGURE 2-9

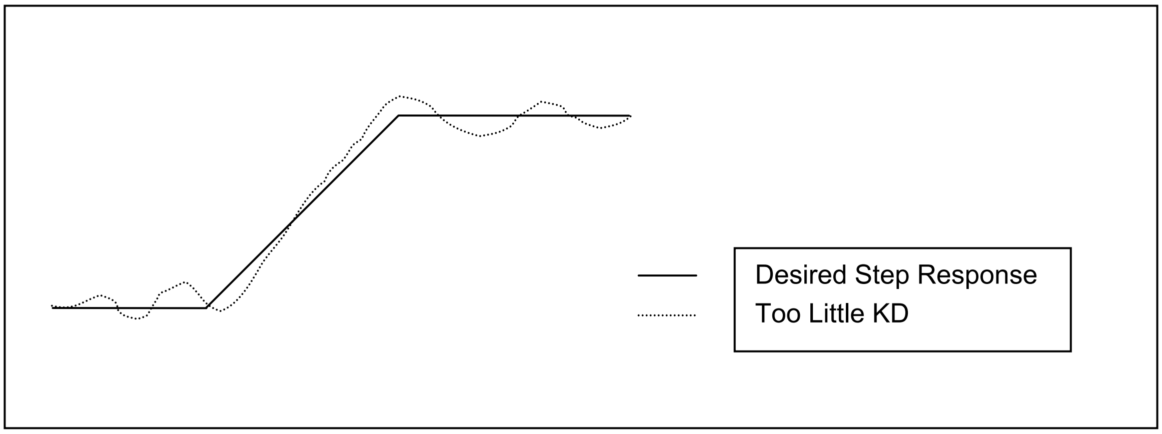


FIGURE 2-10

KP, KI, and KD are the primary parameters of concern when tuning a servo system. Once the optimum values for these variables have been determined, you can adjust some of the secondary parameters that will help fine tune your system's performance. These other variables are described in the subsequent steps.

The KV variable is the velocity feedforward coefficient, and compensates for friction that is proportional to velocity. Unlike KP, KI, and KD, which have to wait for system error before responding, the KV variable has an immediate effect on the commanded move, and is a gain applied to the current velocity. KV makes the system more responsive, and by increasing this term, the "Following Error" of the system's response can be minimized. However, too large of a value may result in unstable behavior after command velocity changes.

The values for KV range from 0.00 to 32767.00.

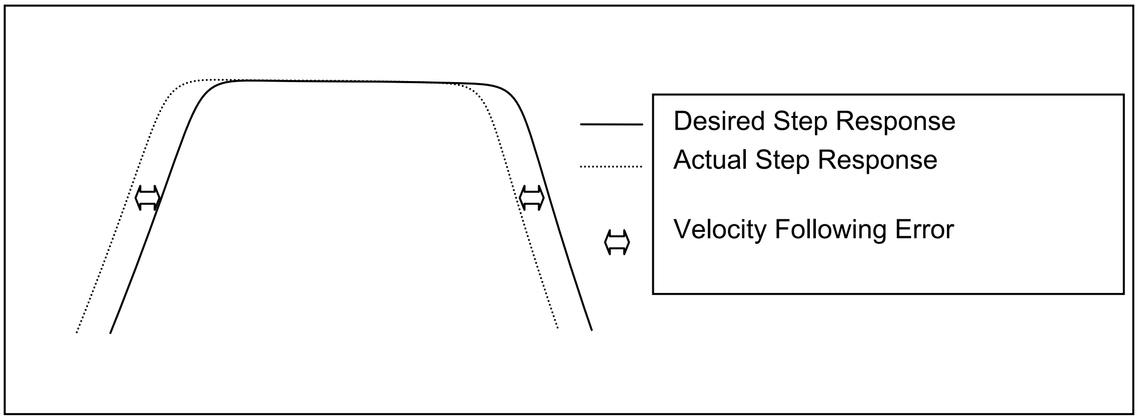


FIGURE 2-11

The KA variable is the acceleration feedforward coefficient, and compensates for inertia. Like KV, the KA variable does not operate on system error, and is applied as a gain to the current acceleration and deceleration. KA determines how closely the system follows the desired acceleration and deceleration portions of the motion profile. Increasing this term reduces the following error occurring during acceleration and deceleration of the system, but if KA is too large, instability may occur.

The values for KA range from 0.00 to 32767.00.

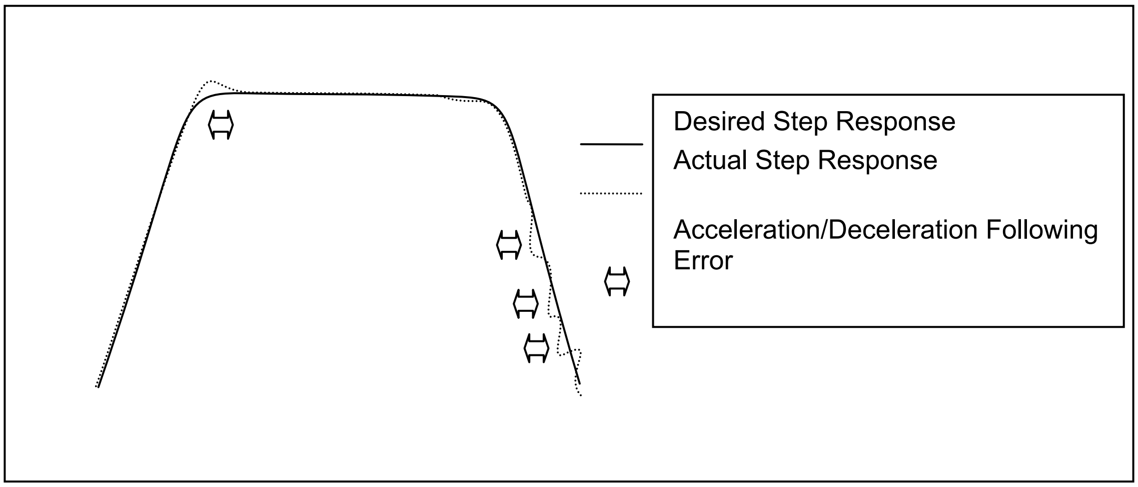


FIGURE 2-12

The KF variable is the friction offset coefficient, and compensates for static friction. The KF variable does not operate on system error, and is applied to all commanded moves. KF increases all portions of the motion profile. If KF is too large, instability may occur.

The values for KF range from 0.00 to 32767.00.

The block diagram below describes the feedback loop that is taking place in the servo system:

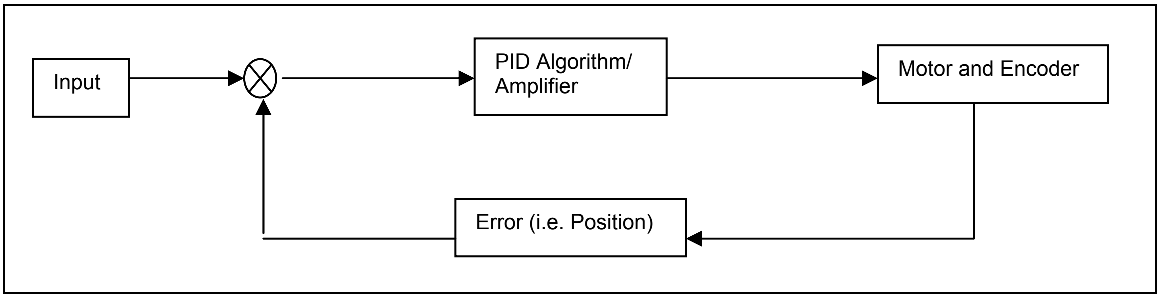


Figure 2-13 Feedback Loop

To verify that your motor is tuned properly, send the commands LP0;CL1; and check the shaft of the motor to make sure it is stiff. If there is play in the motor shaft when you turn it, then you may have to re-adjust your PID filter.

Once you are satisfied with the static holding torque you could check for position error. Send the command "AC100000;VL5000;MR64000;GO;". With a 2000 line encoder this move would be equivalent to 8 revolutions of the motor. After the move is complete check the position error by sending the RE and RP commands for the specific axis you are moving. Compare the difference in the two responses. If they are the same then you are on the right track, if the error is greater than 32768, the controller will disable the PID so that you don't have a runaway motor. In this case major changes to the PID parameters may be required. For minor differences in the encoder and the motor position readings you can fine-tune your PID filter according to the earlier steps.

You may want to save the values for KP, KI, KD, etc., for future reference. These values can be saved in the board's flash memory, so they can be accessed easily on reset or power-up. This can be done by using the APP command. These saved parameters will then be used as the power up default set of values.

2.14. SETTING THE USER DEFAULT CONFIGURATION

There are several parameters that can be defined by the user as default. These parameter values can supersede the factory default values and be stored in flash memory for power-up configuration. Most of these parameters consist of axis specific values; i.e. velocity, acceleration, limit switch, logic sense, etc.

The MAXk comes from the factory with default values for all parameters. For instance, the default value for the velocity of all axes is 200,000 counts per second. (A count is equivalent to a step pulse or one count of an encoder.) In a typical application, when the system is powered up, the main host computer would initialize all of the peripherals, such as the MAXk, sending each of the axes the peak velocity. When the User Definable Default Parameter value is defined, then the velocities of the defined axes will be set accordingly. This feature can greatly simplify the software and initialization process.

Once the values for all of the associated parameters are defined; i.e. velocity, acceleration, PID values, etc. then the APP command is executed to place the values into flash memory. From this point forward these defined values will be used after reset or power-up. The individual parameters can be over-written at anytime by using the associated command; i.e. VL#, AC#, etc. To restore the factory defaults the command RDF is executed. To restore the User Defined Default Parameters the command RDP is executed. The following is a list of parameters that can be defined as part of the User Definable Power-Up Default Parameters.

Over travel limit (soft limit or hard limit); Factory Default = Hard limit

Over travel limit (enabled or disabled); Factory Default = Enabled

Over travel limit polarity (active high or active low); Factory Default = active low

Software based over travel for each axis; Factory Default = disabled

Direction Bit polarity

Acceleration value for each axis; Factory Default = 2,000,000

Trajectory profile for each axis (linear, parabolic, S-curve, custom); Factory Default = Linear

Velocity Peak; Factory Default = 200,000

Velocity Base; Factory Default = 0

User Unit values for each axis; Factory Default = Off

Auxiliary output settle time for each axis; Factory Default = 0

Automatic auxiliary control axis by axis; Factory Default = Off

Encoder Ratio for each axis; Factory Default = 1:1

Encoder Slip tolerance for each axis. (Used for stepper motors); Factory Default = 0

Home Active ‘Low’

Position Maintenance Dead-Band, Hold Gain and Hold Velocity. (Used for stepper systems); Factory Default = 0,0,0

Servo axis unipolar/bipolar output; Factory Default = bipolar

Servo PID values: KP, KD, KI, KO, KV, KA; Factory Default = KP 10, KD 160, KI 1.00, KO 0, KV 0, KA 0

Servo zero value: KO; Factory Default = 0

2.15. POWER SUPPLY REQUIREMENTS

The MAXk Motion Controller Card plugs into the PCI Bus. The MAXk is designed to fit into a standard full size card PCI slot and draws 0.6 Amps from the +5V and 3.3V power supplies of the PC in addition to the possible 1 Amp of +5V provided on the J1 / J2 connectors. For servo models only +12V at 0.1 Amp and -12V at 0.1 Amp are also taken from the PC.

3. COMMUNICATION INTERFACE

3.1. INTRODUCTION

The MAXk is 100% compatible with standard PCs and complies with the PCI Bus Specification, Revision 2.2. The MAXk can be considered a motion co-processor in the PCI computer where it can execute the motion process independent of the host CPU. The application software issues DLL function calls and receives requested data from the driver support DLL. All communication is done between the motion controller and the host PC via the device driver and the DLL. (See also Functional and Data Flow Info Diagrams Figure 3-1.)

3.2. PCI INTERFACE

The PCI interface to the controller consists of a 32 bit register and 64 Kbytes of shared memory.

After the host system BIOS has executed its PCI resource allocation functions the controller's PCI configuration registers will contain the following information:

PCI Register Offset (HEX)	Register Contents (32 bits)
0x000:	The Device ID and Vendor ID. (0006 1057)
	This register identifies the vendor that supplies the PCI bridge chip.
0x018:	The base address of the controller's dual port RAM.
0x020:	The base address of the controller's memory mapped I/O registers.
0x02C:	The controller's Subsystem ID and Subsystem Vendor ID. (0010 + NNNN 160C)

This register identifies the PCI card as an Oregon Micro System controller, with NNNN (see also Figure 2-1) being the board number set on the MAXk card.

0x03C:	Bits 0 through 3 contain the controller's IRQ assignment.

The shared RAM (also called dual port RAM) is used to pass data from the controller to the host computer. The 32 bit word, at offset zero in dual port RAM, contains a binary image of the controller's configuration dip switches. Bits 0 thru 2 contain the controller number selection.

The shared RAM is used extensively in the MAXk, including status flags, text commands, interrupt notification and various registers. The remainder of the RAM is used for axis and encoder position data, velocity profile and servo tuning information to the host.

3.3. PCI COMMUNICATION THEORY

As shown in the simplified diagram below (Figure 3.1), communication between the MAXk controller and the application is via the device driver and its associated driver support DLL.

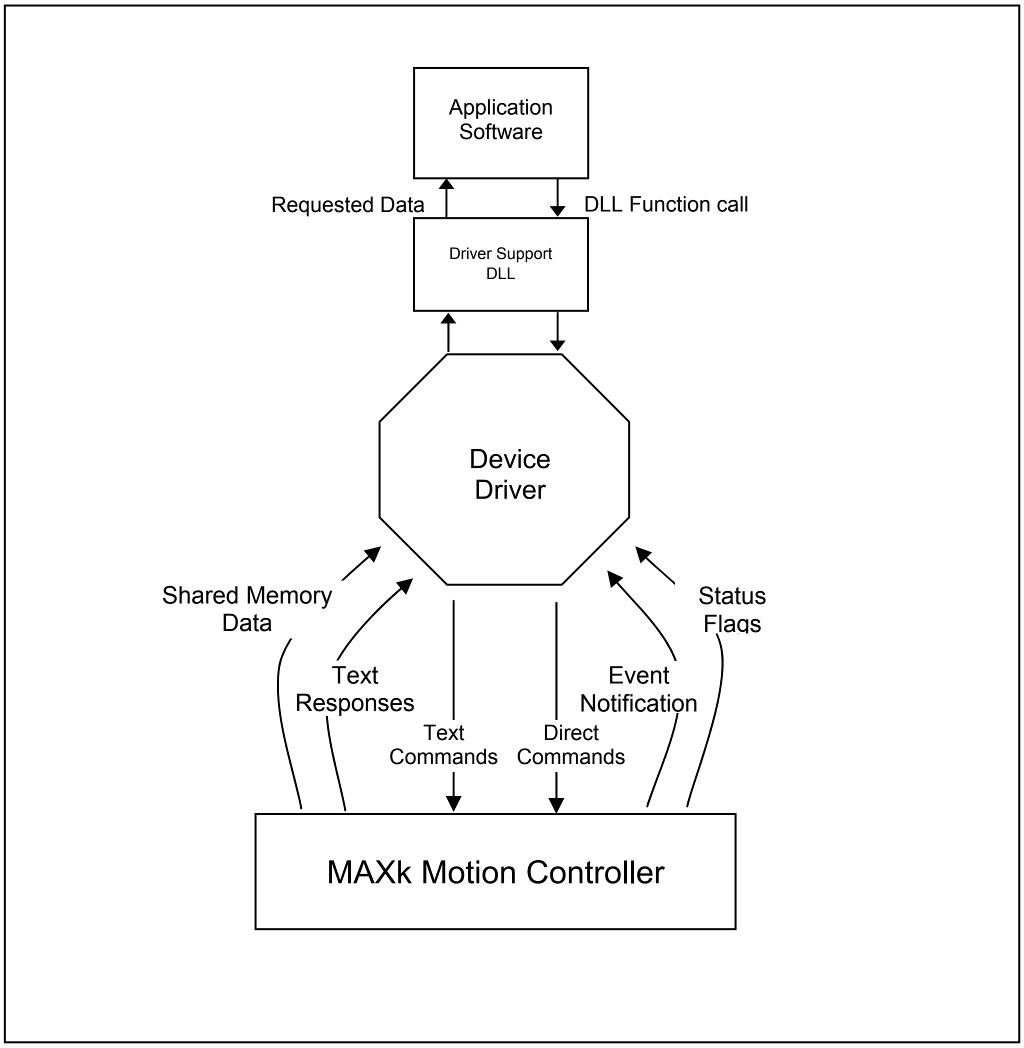


Figure 3-1 FUNCTIONAL COMMUNICATIONS FLOW

Simplified Data Dictionary of Figure 3.1.

Event Notification = {New status flag data available, Query command text response available}

Direct Commands = {Kill all motion}

Text Commands = {ASCII controller command strings}

Status Flags = {

Axis done flags = {X, Y, Z, T, U, V, R, S},

Axis over-travel flags = {X, Y, Z, T, U, V, R, S},

Axis encoder slip flags = {X, Y, Z, T, U, V, R, S},

Command error flag,

Axis notification flags = {W, K}}

Text Responses = {Query command (RP, RE…) ASCII response strings}

Shared Memory Data = {Axis motor position data, axis encoder position data, velocity profile data, and servo tuning data, axis status flags W & K}

Requested Data = {Null terminated ASCII text response string, velocity profile data, servo tuning data, axis done flags, axis over-travel flags, axis encoder slip flags, command error status}

3.4. COMPARISON OF PREVIOUS OMS ARCHITECTURE TO MAXk

OMS motion controllers previously used hardware registers for status, slip, done and over travel limits. The MAXk uses the Power PC's Message unit, in combination with reserved storage regions in the common memory area, to accomplish these functions.

Bits in the 32 bit Out Bound Door Bell Register (ODBR) are used to pass status and flag data to the Host. Bits in the ODBR register are defined as follows:

3.4.1. STATUS DONE FLAG DATA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Axis S	Axis R	Axis V	Axis U	Axis T	Axis Z	Axis Y	Axis X

Done is signaled by setting the appropriate done bits in the Out Bound Door Bell Register (ODBR). The host responds to the door bell interrupt by reading the Door Bell register and capturing the done flag data. The done flags are cleared when the host writes ONES to set the bits.

3.4.2. OVERTRAVEL

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Axis S	Axis R	Axis V	Axis U	Axis T	Axis Z	Axis Y	Axis X

An **Over Travel** condition is signaled by setting the appropriate bit in the ODBR register and the host will respond to the Door Bell interrupt by reading the ODBR and capturing the over travel status. The over travel bits will be cleared when the host writes ONES to set the bits. Note that the over travel bits in the door bell register are used to notify the host that an over travel axis has been detected. The actual LIMIT switch status is obtained by reading the current limit switch status from the common memory area.

3.4.3. SLIP FLAG

Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
Axis S	Axis R	Axis V	Axis U	Axis T	Axis Z	Axis Y	Axis X

Encoder slip will be signaled by setting the appropriate bit in the door bell register. The Host will respond to the door bell interrupt by reading the door bell register and capturing the encoder slip status data. The slip bits will be cleared when the host writes ONES to set the bits.

3.4.4. COMMAND ERROR

The command error bit is ODBR bit 24 and it is cleared when the host writes a ONE to it.

3.4.5. AXES NOTIFICATION FLAGS

Bit 26	Bit 25
Axis K	Axis W

The axis nofication flags for the axes W and K in the ODBR indicate a done/overtravel/slip set bit in its axis status flag register. This keeps the MAXk backward compatible to the MAXp driver.

3.4.6. ASCII COMMAND RING BUFFER

Command characters strings from the host are transferred to the controller by placing it in the ASCII Command Ring Buffer and updating the buffer insert pointer.

3.4.7. ASCII RESPONSE RING BUFFER

This ring buffer is resident in the Common Memory area region and the Message Unit's Outbound Message Register 0 (OMR). Data is sent from the controller to the host by placing data into the ring buffer and then using OMR to notify the host that data is available.

3.5. DYNAMIC LINK LIBRARY

OMS provides a Motion Control DLL for Windows NT, Windows 2000, Windows XP & Windows 7 32-bit operating systems. These DLL's function with the drivers to provide a rich set of commands for use in various motion control environments.

The latest versions of the DLL's are included on a CD-ROM with the shipment of the MAXk Controller card. Also included on a separate CD is a .PDF version of the User Manual. The .PDF version of the MAXk User manual and the support software is also available from the OMS Motion, Inc. website (http://www.omsmotion.com/).

The omsMAXkmc.dll is the Motion Control Dynamic Link Library that provides a means for C, C++ and Visual Basic applications, running under Windows NT, to communicate with Oregon Micro Systems PCI motion controllers. The DLL supplies a set of functions that provide "single function call" solutions to some of the more common motion control problems. For added flexibility other functions are also provided to allow the application to interact with the controller at its most basic text command string and text response string level. The document omsMAXkmc.txt or omsMAXkmc.doc supplies a description of each function, function calling conventions, and code samples. These are available on the support software CD-ROM or on our web site.

3.6. MAXk - PCI ADDRESS SPACE MEMORY / REGISTER MAP

The MAXk family of motion controllers utilizes 0xe010 (57360) bytes in the PCI Controller/Host shared memory address space. The MAXk controller uses a base address of 0x01F0_0000 to access this address space.

Table 3-1 MAXK - PCI SHARED ADDRESS SPACE MAPPING

Byte Offset	Byte Offset (Hex)	Byte length	Description
The following 8 words contain axis motor positions. It is updated each motor update cycle.			
0	0x0	4	X axis motor position
4	0x4	4	Y axis motor position
8	0x8	4	Z axis motor position
12	0xc	4	T axis motor position
16	0x10	4	U axis motor position
20	0x14	4	V axis motor position
24	0x18	4	R axis motor position
28	0x1c	4	S axis motor position
The following 8 words contain the axis encoder positions. It is updated each update cycle.			
32	0x20	4	X axis encoder position
36	0x24	4	Y axis encoder position
40	0x28	4	Z axis encoder position
44	0x2c	4	T axis encoder position
48	0x30	4	U axis encoder position
52	0x34	4	V axis encoder position
56	0x38	4	R axis encoder position
60	0x3c	4	S axis encoder position
The following word contains the axis limit status bits. It is updated each update cycle.			
64	0x40	4	Limit Switch status word
The following word contains the axis home sensor status bits. It is updated each update cycle.			
68	0x44	4	Home Switch status word
The following word contains the controller firmware status flags. It is updated as events occur.			
72	0x48	4	Firmware State flags
The following word is a direct command mechanism that bypasses the text command buffer.			
76	0x4c	4	Direct Command Mail Box
The following 17 words contain a memory region used to capture coherent snapshots of axis position.			
80	0x50	4	Position Request Mail Box
84	0x54	4	X axis motor position
88	0x58	4	Y axis motor position
92	0x5c	4	Z axis motor position
96	0x60	4	T axis motor position
100	0x64	4	U axis motor position
104	0x68	4	V axis motor position
108	0x6c	4	R axis motor position
112	0x70	4	S axis motor position
116	0x74	4	X axis encoder position
120	0x78	4	Y axis encoder position
124	0x7c	4	Z axis encoder position
128	0x80	4	T axis encoder position
132	0x84	4	U axis encoder position

Table 3-1 MAXk - PCI SHARED ADDRESS SPACE MAPPING (CON'T)

Byte Offset	Byte Offset (Hex)	Byte length	Description
136	0x88	4	V axis encoder position
140	0x8c	4	R axis encoder position
144	0x90	4	S axis encoder position
The following word is used to coordinate the sending of text responses from the controller to the host.			
148	0x94	4	Message semaphore
152	0x98	4	Reserved
The following word contains the state of the 16 general purpose I/O bits, updated each update cycle.			
156	0x9c	4	General Purpose I/O bits status
160	0xa0	12	Reserved
The following 2 words contain axis motor positions. It is updated each motor update cycle.			
172	0xac	4	W axis motor position
176	0xb0	4	K axis motor position
The following 2 words contain the axis encoder positions. It is updated each update cycle.			
180	0xb4	4	W axis encoder position
184	0xb8	4	K axis encoder position
The following 4 words contain a memory region used to capture coherent snapshots of axis position.			
188	0xbc	4	W axis motor position
192	0xc0	4	K axis motor position
196	0xc4	4	W axis encoder position
200	0xc8	4	K axis encoder position
204	0xcc	4	W axis status flags
208	0xd0	4	K axis status flags
212	0xd4	28	Reserved
The following memory region contains various data transfer buffers.			
240	0xf0	4	ASCII Command Buffer insert index
244	0xf4	4	ASCII Command Buffer process index
248	0xf8	4	ASCII Response Buffer insert index
252	0xfc	4	ASCII Response Buffer process index
256	0x100	1024	ASCII Command Ring Buffer
1280	0x500	1024	ASCII Response Ring Buffer
2304	0x900	2048	Utility transfer buffer
4352	0x1100	1088	Reserved
5440	0x1540	552	Real-Time Position Capture Data
5992	0x1768	5912	Reserved
The following memory region contains analog I/O data.			
11904	0x2e80	4	X axis DAC output
11908	0x2e84	4	Y axis DAC output
11912	0x2e88	4	Z axis DAC output
11916	0x2e8c	4	T axis DAC output
11920	0x2e90	4	U axis DAC output
11924	0x2e94	4	V axis DAC output
11928	0x2e98	4	R axis DAC output
11932	0x2e9c	4	S axis DAC output
11936	0x2ea0	4	Coherent X axis servo DAC output
11940	0x2ea4	4	Coherent Y axis servo DAC output
11944	0x2ea8	4	Coherent Z axis servo DAC output

Table 3-1 MAXk - PCI SHARED ADDRESS SPACE MAPPING (CON'T)

11948	0x2eac	4	Coherent T axis servo DAC output
11952	0x2eb0	4	Coherent U axis servo DAC output
11956	0x2eb4	4	Coherent V axis servo DAC output
11960	0x2eb8	4	Coherent R axis servo DAC output
11964	0x2ebc	4	Coherent S axis servo DAC output
11968	0x2ec0	4	Coherent W axis servo DAC output
11972	0x2ec4	4	Coherent K axis servo DAC output
11976	0x2ec8	4	W axis DAC output
11980	0x2ecc	4	K axis DAC output
11984	0x2ed0	1616	Reserved
The following memory region contains absolute encoder data.			
13600	0x3520	4	X axis absolute encoder position
13604	0x3524	4	Y axis absolute encoder position
13608	0x3528	4	Z axis absolute encoder position
13612	0x352c	4	T axis absolute encoder position
13616	0x3530	4	U axis absolute encoder position
13620	0x3534	4	V axis absolute encoder position
13624	0x3538	4	R axis absolute encoder position
13628	0x353c	4	S axis absolute encoder position
13632	0x3540	4	W axis absolute encoder position
13636	0x3544	4	K axis absolute encoder position
13640	0x3548	43716	Reserved

Table 3-2 MAXk - Limit switch status word (word access offset 0X40)

Bit	Function
Byte access offset 0x43	
00	X axis negative limit sensor
01	Y axis negative limit sensor
02	Z axis negative limit sensor
03	T axis negative limit sensor
04	U axis negative limit sensor
05	V axis negative limit sensor
06	R axis negative limit sensor
07	S axis negative limit sensor
Byte access offset 0x42	
08	X axis positive limit sensor
09	Y axis positive limit sensor
10	Z axis positive limit sensor
11	T axis positive limit sensor
12	U axis positive limit sensor
13	V axis positive limit sensor
14	R axis positive limit sensor
15	S axis positive limit sensor
Byte access offset 0x41	
16	W axis negative limit sensor
17	K axis negative limit sensor
18	Not used
19	Not used
20	Not used
21	Not used
22	Not used
23	Not used
Byte access offset 0x40	
24	W axis positive limit sensor
25	K axis positive limit sensor
26	Not used
27	Not used
28	Not used
29	Not used
30	Not used
31	Not used

Table 3-3 MAXk - Home Switch Status Word (Word Access Offset 0X44)

Bit	Function
Byte access offset 0x47	
00	X axis home sensor
01	Y axis home sensor
02	Z axis home sensor
03	T axis home sensor
04	U axis home sensor
05	V axis home sensor
06	R axis home sensor
07	S axis home sensor
Byte access offset 0x46	
08	W axis home sensor
09	K axis home sensor
10	Not used
11	Not used
12	Not used
13	Not used
14	Not used
15	Not used
Byte access offset 0x45	
16	Not used
17	Not used
18	Not used
19	Not used
20	Not used
21	Not used
22	Not used
23	Not used
Byte access offset 0x44	
24	Not used
25	Not used
26	Not used
27	Not used
28	Not used
29	Not used
30	Not used
31	Not used

Table 3-4 MAXk CONTROLLER FIRMWARE STATUS FLAGS (WORD ACCESS OFFSET 0x48)

Bit	Function
00	Controller application code not downloaded to RAM.
01	Controller application code is initializing.
02	Controller application code is running.
03	Not used
04	Not used
05	Not used
06	Not used
07	Not used
08	Application stored in flash memory has a check sum error.
09	A programming error occurred while storing the application code in flash memory.
10	Not used
11	Not used
12	A checksum error was detected in the power up default parameter archive.
13	A programming error occurred while storing parameters in the power up default parameter archive.
14	A checksum error was detected in the alternate parameter archive.
15	A programming error occurred while storing parameters in the alternate parameter archive.
16	The power up default parameter set has been loaded into working memory.
17	The alternate parameter set has been loaded into working memory.
18	The factory default parameter set has been loaded into working memory.
19	Not used
20	Not used
21	Not used
22	Not used
23	Not used
24	Not used
25	Not used
26	Not used
27	Not used
28	Not used
29	Not used
30	Not used
31	Not used

NOTE: If the firmware state register contains 0xFFFF_FFFF then the controller has not completed power up initialization.

3.7. REAL-TIME POSITION CAPTURE

The position capture commands control the real-time recording of axis position data and the management of the captured position data. The captured position data includes the axis, the positive edge I/O bits, the negative edge I/O bits, the home and encoder home events, and the encoder position of the axis. The position data is captured when the conditions specified for the input bit are met. The capture conditions for the home switch and general purpose input bits can be a rising/positive edge, a falling/negative edge, or the event can be both the rising/positive and the falling/negative edge so data is captured on any transition of the input bit. The real-time position capture feature is only available on an axis with incremental encoders. See the MAX family command reference manual for more details on the real-time position capture feature.

The MAXk controller has a ring buffer in PCI shared memory, which is used to transfer the real-time position capture data to the host. When a capture event is recorded by the motor update cycle routine, it transfers the capture table entry to the shared PCI memory. The host is signaled that the data is available via bit number 1, or hexadecimal value 0x00000002, in the out bound message register 1, which causes INTA to be asserted on the PCI bus. The shared memory for the capture data is implemented as a ring buffer with an insert index that the controller uses to insert data into the shared memory region, and a removal index that the host uses to remove data from shared memory region. The controller places the capture data into the ring buffer at the location specified by the insert index, and advances the insert index. If after being advanced, the insert index equals the removal index, then the controller also advances the removal index. If the controller has to advance the removal index, this means that the host is not removing data fast enough, and capture data was lost by the host. The capture data is available in the shared PCI memory at offset addresses 0x1540 through 0x1767. The format of the capture table data in shared PCI memory is defined in table 3-5 below.

Table 3-5

REAL-TIME POSITION CAPTURE PCI SHARED MEMORY (WORD ACCESS OFFSET 0X1540)

Byte Offset	Byte Offset (Hex)	Byte length	Description
5440	0x1540	1	Controller insert index
5441	0x1541	1	Host removal index
5442	0x1542	550	Table entries (10 bytes per entry, and 55 entries)

The number of entries can be greater than one for each axis, if capture events occur on back to back motor update cycles, and if the host does not collect the data fast enough. The format of each table entry is defined in table 3-6 below.

Table 3-6 REAL-TIME POSITION CAPTURE TABLE ENTRY

Byte Offset	Byte Offset (Hex)	Byte length	Description
0	0x00	4	Encoder position, offset 0x00 contains MSBs, offset 0x03 contains LSBs
4	0x04	1	Axis (X = 0, Y = 1, etc.)
5	0x05	1	Home event bits: 0x01 = Positive edge home switch 0x02 = Encoder home event 0x04 = Negative edge home switch
6	0x06	2	Positive edge I/O bits, offset 0x06 contains MSBs, and offset 0x07 contains LSBs
8	0x08	2	Negative edge I/O bits, offset 0x08 contains MSBs, and offset 0x09 contains LSBs

A value of 1 for a given bit indicates that it triggered the capture event. A value of 0 for a given bit means it did not trigger the capture event.

The motion controller contains a PowerPC processor, which writes the data in the shared memory in big endian format. If the host processor is not a big endian processor, then appropriate byte swapping to correct for endian differences must be performed by the host processor when accessing the shared memory data.

This page is intentionally left blank.

4. CONTROL SIGNAL INTERFACE

4.1. INTRODUCTION

The MAXk family of motion controllers is available in configurations from one to ten axes to manage combinations of servo and step motor systems. Each MAXk 100-pin connector incorporates half of the overall possible I/O. The first 5 axes and half of the digital and analog I/Os are accessable on connector J1. Models with 6 or more axes provide the remaining axes and I/Os on connector J2 which is wired to J3 and J4. The MAXk default configuration is as an open loop stepper controller for the number of axes ordered.

The MAXk controller fully meets the PCI Bus Specification, Revision 2.2 and plugs directly into a PCI slot in a computer motherboard.

4.2. LIMIT INPUTS

To facilitate system safety, TTL inputs for +/- limit conditions are provided for each axis. Limits may be activated by mechanical switches using contact closures or other suitable active switches, such as a Hall Effect switch or opto-isolator that connects to ground.

If the motor travels beyond its allowable limits and trips the switch, the limit condition removes any further excitation in the limit direction from the affected axis. (Servo Motor systems should be designed for safety, i.e. to have electrical braking to stop them). The limit switch active signal state can be selected with the LT command on an axis by axis basis. The behavior of the limit functionality can be set for the axis to decelerate to a stop or to stop without deceleration when a limit condition occurs.

Reference MAX Command Manual (): LM and LT commands.

4.3. HOME INPUTS

To facilitate positioning of an axis to a known reference position, a TTL home input is provided for each axis. For axes using an encoder, the home input can be used in conjunction with the index signal of the encoder. The logic of the encoder signals Phase A/Phase B/Index that constitutes a true home condition is programmable.

Reference MAX Command Manual (http://www.omsmotion.com/): EH, HM, HR, HT, KM, and KR commands.

4.4. GENERAL PURPOSE DIGITAL I/O

There are 16 general purpose digital lines that can be individually configured as either a TTL input or an active drive TTL output. Commands are provided for setting the I/O direction of the lines, setting the state of the outputs and reading the current state of the I/O lines. Inputs can be used to control loops, qualify motion or signal an event.

Reference MAX Command Manual (http://www.omsmotion.com/): BD, BH, BL, BW, BX, IOK, and SW commands.

4.5. ANALOG I/O

4.5.1. ANALOG INPUTS

The 4 general purpose analog inputs are available to read +/-10V values via 16-bit analog to digital converters. These can be used to provide input from analog sensors to application software. Analog inputs can also be configured to provide a velocity override input.

Reference MAX Command Manual (http://www.omsmotion.com/): AI, AO and VOA commands.

4.5.2. ANALOG OUTPUTS

The 12 analog outputs provide +/- 10V outputs via the 16-bit digital to analog converters (DAC). Each servo axis requires a dedicated DAC. There are twelfe DACs and a maximum of ten axes. So there is always at least one general purpose analog output available per connector. For every available axis that is not being used as a servo motor, there is a general purpose analog output available. Each DAC has a +/- 15mA maximum output current.

4.6. CONTROL OUTPUT

The MAXk is configured at the factory to control open loop stepper motors. Upon installation, each axis can be configured for servo motors, open loop steppers, stepper motors with encoder feedback or a combination thereof. The servo output may be either unipolar analog (0/+10V) or bipolar analog (-10/+10 V). Each axis configured as a servo motor uses one analog output.

Step pulse and direction outputs are active drive TTL level output signals which will wire directly into most driver inputs.

Auxiliary outputs are active drive TTL outputs.

Each step, direction and auxiliary output has a +/- 20mA maximum output current.

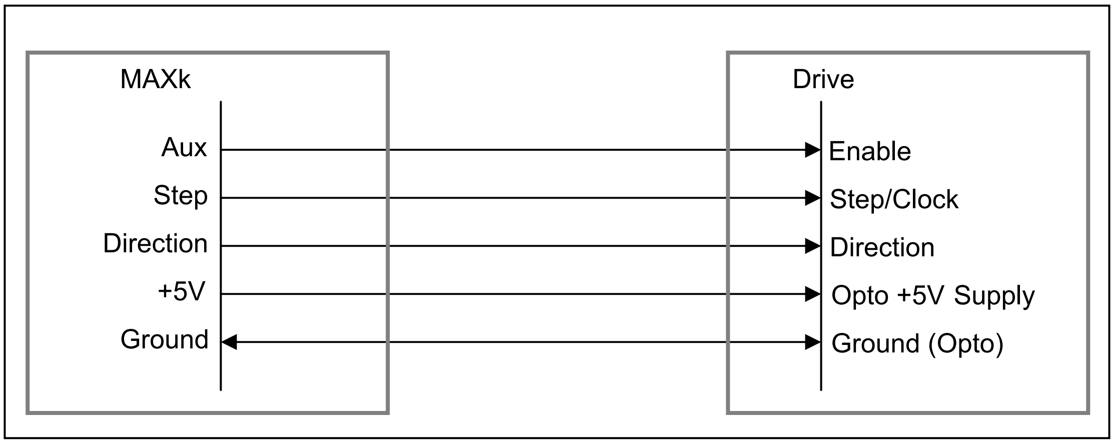


Figure 4-1CONNECTION TO STEP DRIVES

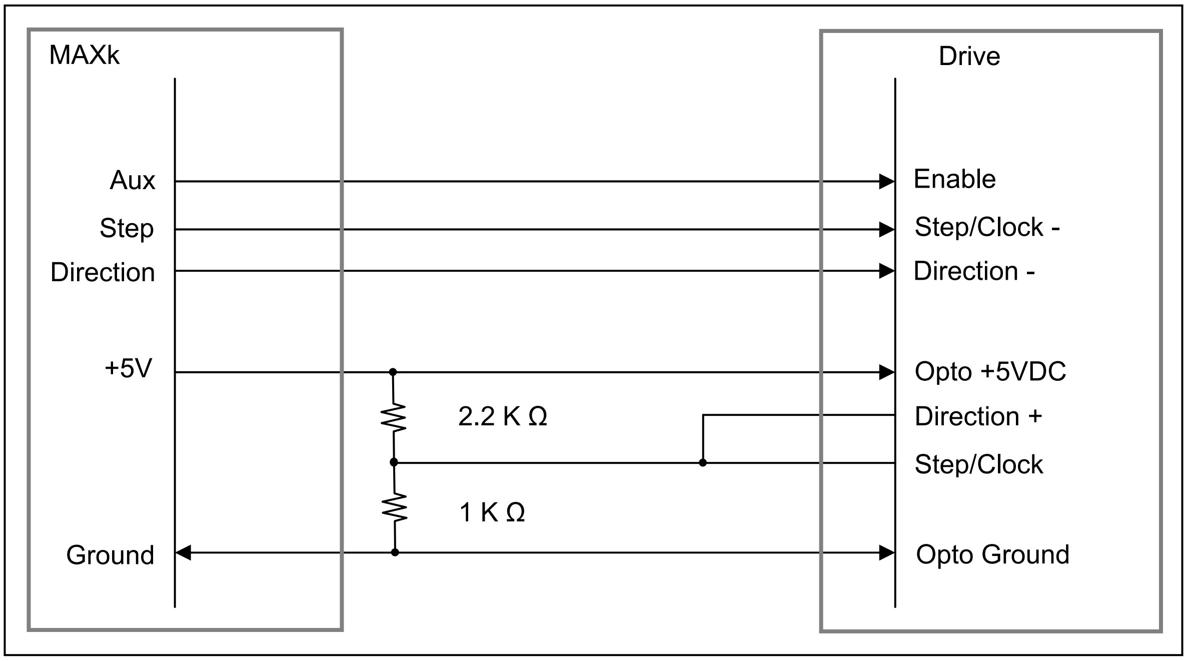


Figure 4-2 CONNECTION TO STEP DRIVES WITH DIFFERENTIAL DRIVE INPUTS

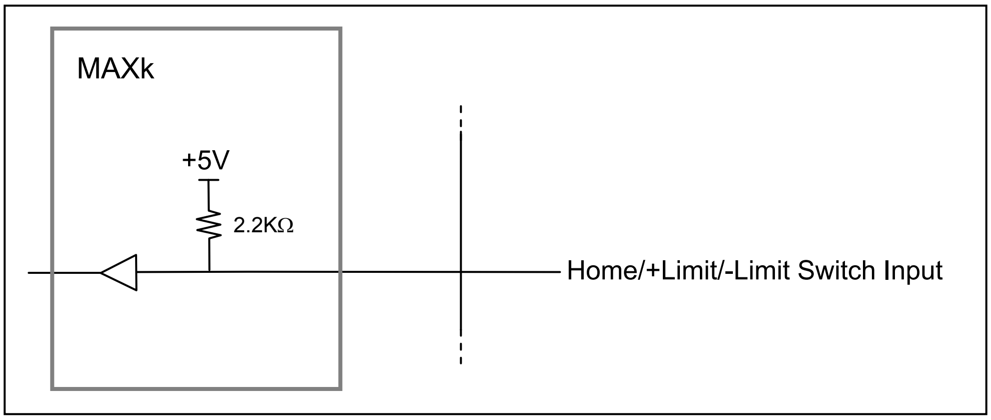


Figure 4-3 HOME AND LIMIT INPUT WIRING DIAGRAM

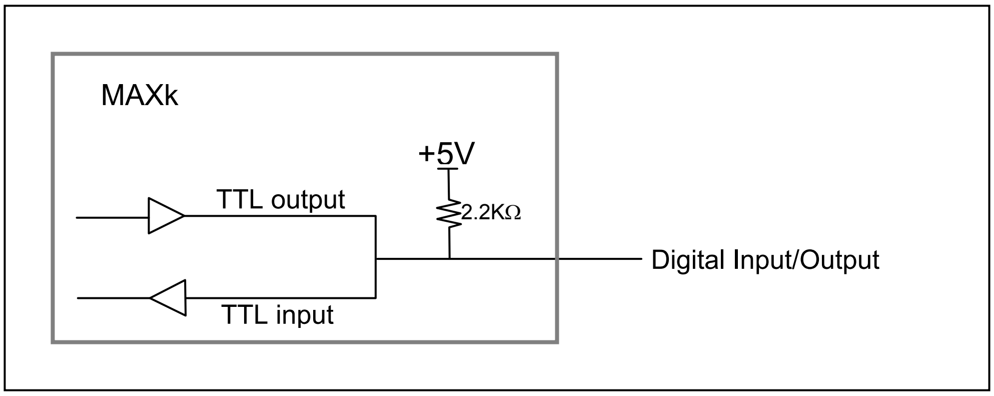


Figure 4-4 GENERAL PURPOSE I/O WIRING DIAGRAM

4.7. ENCODER FEEDBACK

Incremental encoder feedback is provided for all axes. Encoder feedback is required for each servo axis. Its use is optional for stepper axes. The MAXk encoder feedback accepts quadrature pulse inputs from high resolution encoders at rates up to 16 MHz (after quadrature detection). When used with stepper motors, the encoder monitors the actual position through the encoder pulse train. On servo axes it continuously provides input to calculate the position error, adjust for it through the PID filter, and change the output accordingly. The stepper axes can monitor the error and correct and maintain the position after the move is finished.

The encoder input can also be used as an independent feedback source or, in the encoder tracking mode, to mimic an activity. All modes are capable of slip or stall detection and encoder tracking with electronic gearing. These options are selectable by the user through software commands.

The MAXk is compatible with virtually any incremental encoder which provides quadrature outputs. Times four quadrature detection is used to increase resolution. This means that an encoder rated for 1000 counts (or lines) per revolution will result in 4000 counts of quadrature encoded input for each encoder shaft revolution. The inputs are compatible with encoders that have single ended or differential TTL outputs. The MAXk has differential line receivers to accommodate encoders with differential outputs. Encoders with single-ended outputs can wire the signals to the plus(+) side of the differential line receiver and 1.5V bias the minus(-) side.

4.8. HOME PROCEDURES

Two logical input functionalities are provided to synchronize the physical hardware with the MAXk controller; i.e. put the controlled motor in the home position.

The home switch input is a TTL input signal. If current limiting is required, it should be done externally to the board. Contact OMS Motion, Inc. technical support for assistance.

The MAXk home switch input can be used to physically home a mechanical stage. When this functionality is used the axis position counter will be reset to a selected value when the switch is activated. At this point the MAXk can either ramp the axis to a stop or stop the axis immediately. The control of the direction of travel, the logic active state and the response to the active switch are controlled through commands.

The other homing method on the MAXk uses the home switch and the encoder signals to home a motor. When using the Home Encoder (HI) mode, the homing logic is used with these input signals. The home position consists of the logical AND of the encoder index pulse, the home switch input, and a single quadrant from the encoder logic. The home switch and encoder should be positioned relative to each other in such a way that there is only a single location in the entire travel of the axis that creates a true condition for the defined home logic. The HT and EH commands can be used to create different patterns for the home logic, including the option to ignore an encoder phase signal. The default home logic expressed in Boolean terms is:

Home = Phase +A * Phase -B * Index * Home Switch (Default)

It is necessary that the configured quadrant occurs within the index pulse as provided by the encoder for this logic to function properly. The encoder counter (read by a RE/RI commands) must increase for positive moves or the system will oscillate due to positive feedback. For other options, please contact Technical Support.

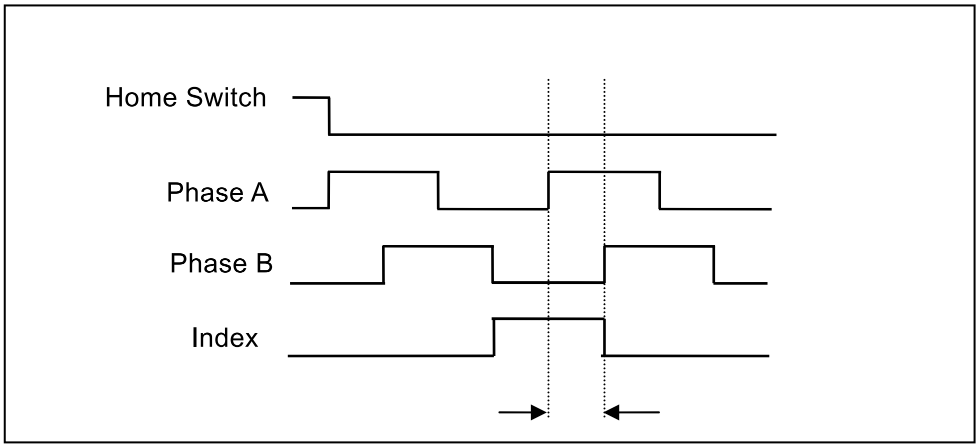


Figure 4-5 ENCODER HOMING STATE DETECTION

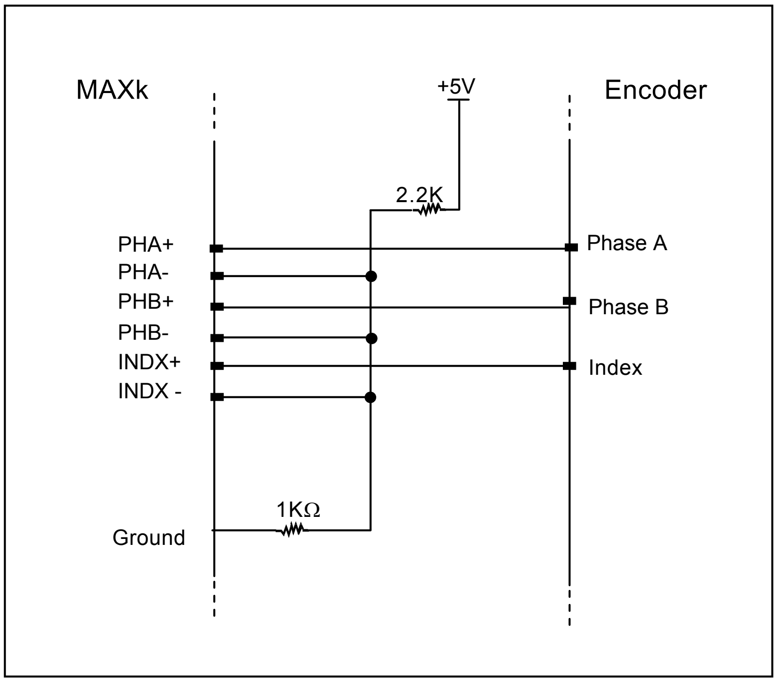


Figure 4-6 ENCODER WIRE DIAGRAM FOR SINGLE-ENDED INPUT SIGNALS

NOTE: The differential receiver has an internal bias on the negative input and may not need the external bias. For better reliability external bias is recommended.

4.9. ABSOLUTE ENCODERS WITH SSI

The MAXk comes with two axes of configurable absolute encoders with SSI (Synchronous Serial Interface) technology. By default the X and Y axes will have up to 12 bits of resolution of absolute encoding. SSI encoder feedback for each axis with resolution up to 32 bit is available opon request. The MAXk provides a differential clock output through the encoder index port to deliver clocking to an absolute encoder. The data signal is wired to encoder phase A. The data clocking can be configured for the frequencies 31,250Hz, 62,500Hz, 125,000Hz, 250,000HGz, 500,000Hz, 1MHz, 2MHz, and 4MHz.

4.9.1. CONFIGURATION EXAMPLES

The following are two examples on how to configure the MAXk for absolute encoding. The first case is the standard MAXk with two absolute encoders with up to 12 bits resolution. For this example, the X axis is 12 bits resolution with a clock frequency at 125,000Hz, and the Y axis is 9 bits resolution with a clock frequency of 250,000Hz.

```
AX;
PSE;
ECA12,125000;
AY;
PSE;
ECA9,250000;
```

The second example calls for five absolute encoders, two axes at 16 bits resolution with a clock frequency of 125,000Hz, one axis at 24 bits resolution with a clock frequency of 500,000Hz, and two axes at 32 bits resolution at 250000Hz. This example also shows the use of clock sharing with other absolute encoders with the same clock frequency and bits resolution.

```
AX;
PSE;
ECA16,125000;
AY;
PSE;
ECA16,125000;
AZ;
PSE;
ECA24,500000;
AT;
PSE;
ECA32,250000;
AU;
PSE:
ECA32,250000;
```

Below is an example of how the absolute encoder can be connected to the MAXk. This utilizes the IOMAXnet breakout board for easier connectivity to the absolute encoder environment.

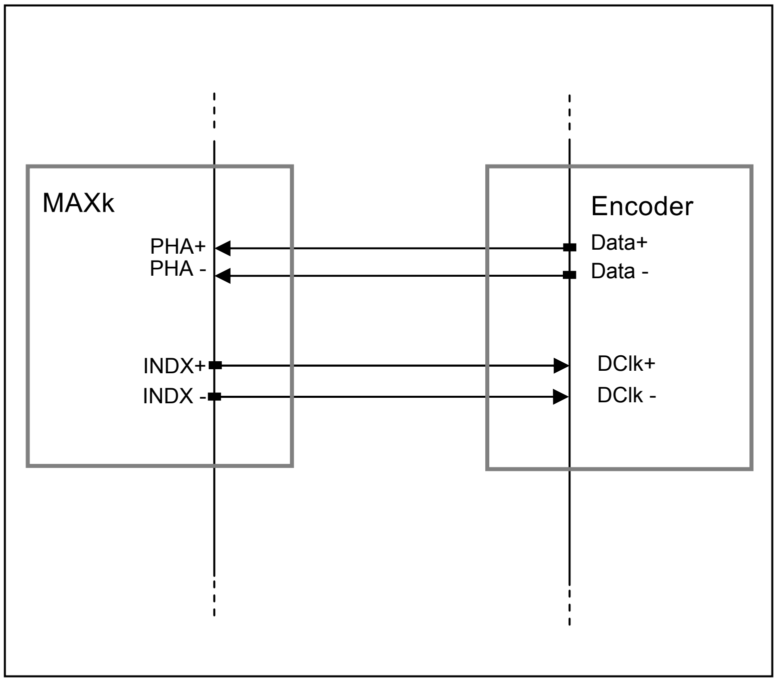


Figure 4-7 SSI ENCODER WIRE DIAGRAM

Table 4-1 OUTPUT CONNECTORs PIN LIST (J1, J2)

J1 – 100-pin connector				**J2 – 100-pin connector**			
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	X Phase +A	51	Y Phase +A	1	V Phase +A	51	R Phase +A
2	X Phase -A	52	Y Phase –A	2	V Phase -A	52	R Phase –A
3	X Phase +B	53	Y Phase +B	3	V Phase +B	53	R Phase +B
4	X Phase -B	54	Y Phase -B	4	V Phase -B	54	R Phase -B
5	X Index +	55	Y Index +	5	V Index +	55	R Index +
6	X Index -	56	Y Index -	6	V Index -	56	R Index -
7	Z Phase +A	57	GROUND	7	S Phase +A	57	GROUND
8	Z Phase -A	58	GROUND	8	S Phase -A	58	GROUND
9	Z Phase +B	59	T Phase +A	9	S Phase +B	59	W Phase +A
10	Z Phase -B	60	T Phase –A	10	S Phase -B	60	W Phase –A
11	Z Index +	61	T Phase +B	11	S Index +	61	W Phase +B
12	Z Index -	62	T Phase –B	12	S Index -	62	W Phase –B
13	U Phase +A	63	T Index +	13	K Phase +A	63	W Index +
14	U Phase -A	64	T Index -	14	K Phase -A	64	W Index -
15	U Phase +B	65	U Index +	15	K Phase +B	65	K Index +
16	U Phase -B	66	U Index -	16	K Phase -B	66	K Index -
17	GROUND	67	GROUND	17	GROUND	67	GROUND
18	X Positive Limit	68	X Negative Limit	18	V Positive Limit	68	V Negative Limit
19	Y Positive Limit	69	Y Negative Limit	19	R Positive Limit	69	R Negative Limit
20	Z Positive Limit	70	Z Negative Limit	20	S Positive Limit	70	S Negative Limit
21	T Positive Limit	71	T Negative Limit	21	W Positive Limit	71	W Negative Limit
22	U Positive Limit	72	U Negative Limit	22	K Positive Limit	72	K Negative Limit
23	GROUND	73	GROUND	23	GROUND	73	GROUND
24	X Home	74	X Direction	24	V Home	74	V Direction
25	Y Home	75	Y Direction	25	R Home	75	R Direction
26	Z Home	76	Z Direction	26	S Home	76	S Direction
27	GROUND	77	GROUND	27	GROUND	77	GROUND
28	T Home	78	T Direction	28	W Home	78	W Direction
29	U Home	79	U Direction	29	K Home	79	K Direction
30	GROUND	80	GROUND	30	GROUND	80	GROUND
31	X Aux	81	X Step	31	V Aux	81	V Step
32	Y Aux	82	Y Step	32	R Aux	82	R Step
33	Z Aux	83	Z Step	33	S Aux	83	S Step
34	**5 Volts**	84	GROUND	34	**5 Volts**	84	GROUND
35	T Aux	85	T Step	35	W Aux	85	W Step
36	U Aux	86	U Step	36	K Aux	86	K Step
37	GROUND	87	GROUND	37	GROUND	87	GROUND
38	IO0	88	IO1	38	IO8	88	IO9
39	IO2	89	IO3	39	IO10	89	IO11
40	IO4	90	IO5	40	IO12	90	IO13
41	IO6	91	IO7	41	IO14	91	IO15
42	GROUND	92	GROUND	42	GROUND	92	GROUND
43	ADC 0	93	ADC 1	43	ADC 2	93	ADC 3
44	GROUND	94	GROUND	44	GROUND	94	GROUND
45	X Servo	95	Y Servo	45	V Servo	95	R Servo
46	GROUND	96	GROUND	46	GROUND	96	GROUND
47	Z Servo	97	T Servo	47	S Servo	97	W Servo
48	GROUND	98	GROUND	48	GROUND	98	GROUND
49	U Servo	99	DAC 0	49	K Servo	99	DAC 1
50	GROUND	100	GROUND	50	GROUND	100	GROUND

4.10. IOMAXnet ADAPTER MODULE

The optional IOMAXnet is an adapter module designed to provide easy connection for each signal of the MAXk and MAXnet. It incorporates two-row terminal blocks. It is used with a 10 foot shielded cable to connect to the MAXk via the 100-pin connector. The +5VDC on the IOMAXnet is supplied by the MAXk.

This supply voltage is intended to be utilized with accessories used in conjunction with the MAXk such as sensors, motor driver modules, etc. and supports a maximum current of 0.5 amps for these purposes.

For controller models with 6 or more axes a second IOMAXnet provides the connectivity for the second 100-pin connector.

Table 4-2

IOMAXnet – Terminal Block Pin-Out							
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
52	Y Phase –A	51	Y Phase +A	2	X Phase -A	1	X Phase +A
54	Y Phase -B	53	Y Phase +B	4	X Phase -B	3	X Phase +B
56	Y Index -	55	Y Index +	6	X Index -	5	X Index +
58	GROUND	57	GROUND	8	Z Phase –A	7	Z Phase +A
60	T Phase –A	59	T Phase +A	10	Z Phase –B	9	Z Phase +B
62	T Phase –B	61	T Phase +B	12	Z Index -	11	Z Index +
64	T Index -	63	T Index +	14	U Phase –A	13	U Phase +A
66	U Index -	65	U Index +	16	U Phase –B	15	U Phase +B
68	X Negative Limit	67	GROUND	18	X Positive Limit	17	GROUND
70	Z Negative Limit	69	Y Negative Limit	20	Z Positive Limit	19	Y Positive Limit
72	U Negative Limit	71	T Negative Limit	22	U Positive Limit	21	T Positive Limit
74	X Direction	73	GROUND	24	X Home	23	GROUND
76	Z Direction	75	Y Direction	26	Z Home	25	Y Home
78	T Direction	77	GROUND	28	T Home	27	GROUND
80	GROUND	79	U Direction	30	GROUND	29	U Home
82	Y Step	81	X Step	32	Y Aux	31	X Aux
84	GROUND	83	Z Step	34	5 Volts	33	Z Aux
86	U Step	85	T Step	36	U Aux	35	T Aux
88	IO1	87	GROUND	38	IO0	37	GROUND
90	IO5	89	IO3	40	IO4	39	IO2
92	GROUND	91	IO7	42	GROUND	41	IO6
94	GROUND	93	ADC 1	44	GROUND	43	ADC 0
96	GROUND	95	Y Servo	46	GROUND	45	X Servo
98	GROUND	97	T Servo	48	GROUND	47	Z Servo
100	GROUND	99	DAC 0	50	GROUND	49	U Servo

For a black and white version of this table, click Table 6-2

5. HOST SOFTWARE

5.1. INTRODUCTION TO MAXk SOFTWARE SUPPORT

A disk containing device drivers, application software, and demonstration code for Pro-Dex-Oregon Micro Systems MAXk family controllers is supplied with the purchase of a MAXk controller. Refer to the text files on the disk for installation instructions and other information.

Some programs on the demo disk that include source code may be adapted for use in application programs that use OMS motion controls. No license is required.

The software is also available on the OMS Motion, Inc.' web page: http://www.pro-dexoms.com/.

This page intentionally left blank

6. SERVICE

6.1. USER SERVICE

The MAXk family of controllers contain no user serviceable parts.

6.2. THEORY OF OPERATION

The MAXk controller uses a PowerPC microprocessor for the core of its design. The highest priority process calculates the desired velocity at the selected update rate with a proprietary algorithm (patent number 4,734,847). This frequency is written to logic on board which generates the pulses for stepper motor control and/or the appropriate voltage levels for Servo Control. The velocity profile and synchronization of each axis is also handled by the PowerPC.

The commands from the PCI computer are temporarily stored in a character buffer until the MAXk can parse them. The command is then executed immediately or routed to separate command queues for each axis. The command queue contains a list of addresses to execute. The argument queue stores the parameters (as applicable) supplied with each command for the axis. A command from the host may be expanded into several commands to the appropriate axis. The GO command, for example, will expand into start, ramp up, constant velocity and ramp down commands. The LS command will save its parameter in the argument queue, the loop count, on a loop stack along with the address of the LS command to be used by the next LE command as a target for a jump command are stored in the command queue. The LE command will decrement the loop count and jump to the most recent LS command providing the loop count has not reached zero. If the loop count has reached zero and it is not nested inside another loop, the queue space will be flagged as available and the next instruction in the queue will be executed.

The communication interface is performed by the MAXk microprocessor. Interrupts from the MAXk to the PCI host are generated by this component. Status of the interrupts and error flags may be read by the host in the status register.

6.3. MAXk FIRMWARE UPGRADE

CAUTION:

The firmware upgrade utility erases the flash memory of the controller during the upgrade process. While every precaution has been taken to recover from any failures, the customer should also take every precaution to provide a stable environment for the upgrade process in order to minimize the chance of an irrecoverable error that would require the board to be sent back to the factory.

Files Required:

MAXKUPG.EXE	Host executable that does the firmware upgrade via PCI bus
MAXK.BIN	The default new controller firmware file
MICROUPG.BIN	Controller firmware for assisting MAXKUPG This file is only required if no firmware is running on controller when MAXKUPG is executed. This can occur when an attempt to do a firmware upgrade has failed or has been interrupted before completion.

MAXKUPG command line parameters:

/c=***n***	***n*** specifies controller number (default is 1) (valid is 1-8)
/f=***n***	***n*** specifies controller firmware filename (default is MAXk.bin)
/e=***n***	***n*** specifies number of blocks to erase (default is 2) (valid is 1-4)
/h	for help – display a usage line (no default)

Description of operation:

MAXKUPG will first verify the existence and the validity of the firmware upgrade file prior to sending any commands to the controller to erase or program flash.

Next MAXKUPG determines whether there is firmware currently executing on the MAXk controller. If firmware is currently executing then MAXkupg proceeds to the next step of the upgrade process. Otherwise, MAXKUPG will upload the microupg.bin file to the controller's RAM and start the controller execution of the microupg firmware. Once MAXKUPG determines that the microupg firmware is executing, MAXKUPG proceeds to the next step of the upgrade process.

Next the flash code blocks are erased. If a failure occurs erasing flash, the program is aborted.

Next the firmware file is sent to the controller and the controller programs the flash to the flash code blocks. The firmware is sent in packets. If any single packet has a transmission error, it will be resent until the transmission is successful or the user aborts the upgrade program.

Next the first code block is re-written with a valid MAXk signature so that the boot block will recognize that valid firmware is loaded in the flash.

Finally, a restart command is sent to the controller, causing the controller to re-boot with the new firmware.

Verify that the controller is running the new firmware by running the MAXKCOMM utility and sending a WY command to the controller. The WY response should indicate the version number of the upgraded firmware.

If the WY response does not reflect the version number of the new firmware, then power cycle the PC and run MAXKCOMM after the PC is re-booted. The new firmware version should now be reflected in the WY response.

If MAXKUPG encounters an error that will not allow it continue, it will either abort the program or continuously retry until the error no longer occurs. In the case of a continuous error that does not clear, the user can abort by entering Control-C (holding down the 'Ctrl' key and pressing 'C'). If an upgrade is interrupted by the user or a power failure, simply run MAXKUPG again.

Examples:	**Description:**
MAXkupg	upgrades flash of controller 1 with file MAXk.bin
MAXkupg /f=newMAXk.bin	upgrades flash of controller 1 with file newMAXk.bin
MAXkupg /c=3	upgrades flash of controller 3 with file MAXk.bin
MAXkupg /e=3 /c=2	upgrades flash of controller 2 with file MAXk.bin, first erasing 3 flash blocks instead of the default of 2
MAXkupg /h	will display a help message

Error Messages:

Unknown operating system type

The operating system was not one that is valid for MAXKUPG.

Operating version detection error

MAXKUPG could not successfully detect the operating system.

Invalid controller selection

The parameter supplied with the /C command line option was not valid. The valid range is 1-8.

Invalid erase blocks selection

The parameter supplied with the /E command line option was not valid. The valid range is 1-4.

Invalid command line option

The character following a / character on the command line was not one of the valid command line options. Valid options are /C, /E, /F.

Device OmsMAXk*n* not found

A handle could not be opened for a MAXk controller number *n*.

Flash upgrade failed

Some failure occurred during the erasing of the flash or the writing of the upgrade file to the flash.

Firmware=*xxxxxxxx* : Unknown Status! Exiting.

MAXKUPG could not determine the state of the firmware executing on the MAXk controller.

Cannot get file handle: *filename*

A problem occurred trying to open the upgrade file named *filename*.

Cannot open file: *filename*

A problem occurred opening the upgrade file named *filename*.

Invalid upgrade file signature!

The upgrade file did not have the required signature of a valid upgrade file.

Upgrade File Seek Error!
An error occurred trying to access the upgrade file.

Could not read serial number!
An error occurred trying to read the serial number from the upgrade file.

Invalid upgrade file for MAXk serial number *nnnnnn*
An upgrade file with a serial number lock installed did not match the MAXk controller found.

Flash write failure on packet: *ppp*
The controller did not accept the upgrade packet number *ppp* due to a flash write failure.

Flash verify failure on packet: *ppp*
The controller did not accept the upgrade packet number *ppp* due to a flash verify failure. If the controller successfully wrote the packet to flash, then it attempts to read it back from flash and compare it to the packet data. This error occurs when the packet data is not equal to the flash data.

CRC failure on packet: *ppp*
The controller did not accept the upgrade packet number *ppp* due to a failure to pass the crc checksum test. The controller does not attempt to write to flash until the crc checksum is successful for the packet.

Unknown failure on packet: *ppp*
The controller did not accept the upgrade packet number *ppp* due to some unknown failure.

NAK on ram upload packet
When MAXKUPG detects that no firmware exists on the MAXk controller, it attempts to write a short program to the controller RAM, using a write and read back verification. This error occurs when the MAXk controller did not accept the packet being written to the controller's RAM. MAXKUPG will retry continuously until the write packet is successful.

NAK on packet verification
When MAXKUPG detects that no firmware exists on the MAXk controller, it attempts to write a short program to the controller RAM, using a write and read back verification. This error occurs when the read back of the packet was unsuccessful. MAXKUPG will retry continuously until the read back is successful.

Data verification failure – retry
When MAXKUPG detects that no firmware exists on the MAXk controller, it attempts to write a short program to the controller RAM, using a write and read back verification. This error occurs when the data packet read back from the controller did not compare equally to the data packet written to the controller. MAXKUPG will attempt to re-send the packet until it is successfully verified or the user aborts MAXKUPG.

Erasing code block n... FAILURE.
An error occurred attempting to erase a block of flash.

WAIT messages:
(NOTE: messages preceded with "WAIT:" will be displayed continuously until the condition indicated is met.)

Waiting for boot code to complete initialization...

This message indicates that the boot code is still performing the hardware initialization functions. This message will be displayed continuously until the boot code completes initialization.

Waiting for Controller to enter Upgrade Mode!

This message indicates that MAXKUPG has requested that the controller enter upgrade mode, but the controller has not yet indicated that is in upgrade mode.

Firmware Status = *xxxx* - Waiting for microupg application to start...

This message indicates that the microupg.bin application that has been successfully uploaded to the MAXk controller's RAM, but has not yet indicated that it has started execution.

INFO messages:

MAXKUPG Version: *n.nn*

Displays the current version of the MAXk firmware upgrade utility.

Firmware Required.

MAXKUPG has determined that there is no application running on the MAXk controller. It will attempt to upload the microupg.bin application to the controller's RAM.

Application Running.

MAXKUPG has determined that an application is running on the MAXk controller. The application could either be a previously existing motion control firmware or it could be the microupg.bin that MAXKUPG has just uploaded to the controller's RAM.

Upgrading flash with *filename*.

MAXKUPG has determined that the controller is ready to start the upgrade.

Signature packet re-sent successfully.

This message indicates the last packet with a valid firmware signature has been successfully sent to the controller flash memory.

RESTARTING with new firmware.

This message indicates that the firmware upgrade was successful and MAXKUPG has sent a command to the controller to restart. This will start the controller running with the new upgraded firmware.

Sending *nnn* packets

This message indicates how many packets are going to be sent to the controller flash when sending the new firmware to the controller.

Uploading *nnn* packets

This message indicates how many packets are going to be sent to the controller RAM when uploading the microupg.bin to the controller.

JUMPING TO Application Start Address.

This message indicates that the microupg.bin has been successfully uploaded to the controller RAM and MAXKUPG is now sending a command to the controller to start execution of the microupg.bin program.

Erasing flash block *n*...success

This message indicates that flash block *n* has successfully been erased.

Serial Number verified!

This message indicates that the upgrade file had a Serial Number lock installed and it was successfully verified that the specified upgrade file was valid for the MAXk controller specified.

APPENDIX A.

LIMITED WARRANTY

The Seller warrants that the articles furnished are free from defect in material and workmanship and perform to applicable, published OMS Motion, Inc., Inc. specifications for one year from date of shipment. This warranty is in lieu of any other warranty express or implied. In no event will Seller be liable for incidental or consequential damages as a result of an alleged breach of the warranty. The liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned f.o.b. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect or unauthorized repair are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for 90 days following date of shipment by Seller of the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to Buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

This page is intentionally left blank

APPENDIX B

TECHNICAL SUPPORT

Pro-Dex, Inc - Oregon Micro Systems , Inc. can be reached for technical support by any of the following methods:

1. Internet E-Mail: mailto:support@pro-dex.com

2. World Wide Web: http://www.pro-dexoms.com/

3. Telephone: 8:00 a.m. - 5:00 p.m. Pacific Standard Time
   (503) 629-8081 or (800) 707-8111

4. Facsimile: 24 Hours
   (503) 629-0688

5. USPS: OMS MOTION, INC., INC.
   15201 NW Greenbrier Parkway
   B-1 Ridgeview
   Beaverton, OR 97006

RETURN FOR REPAIRS

Call Pro-Dex, Inc - Oregon Micro Systems Customer Service at (503) 629-8081 or (800) 707-8111 or E-Mail to mailto:salesor@pro-dex.com.

Explain the problem and we may be able to solve it on the phone. If not, we will give you a Return Materials Authorization (RMA) number.

Mark the RMA number on the shipping label, packing slip and other paper work accompanying the return. We cannot accept returns without an RMA number.

Please be sure to enclose a packing slip with the RMA number, serial number of the equipment, reason for return, and the name and telephone number of the person we should contact if we have further questions.

Pack the equipment in a solid cardboard box secured with packing material.

Ship prepaid and insured to:

OMS MOTION, INC., INC.
15201 NW Greenbrier Parkway
B-1 Ridgeview
Beaverton, OR 97006

This page is intentionally left blank

APPENDIX C
SPECIFICATIONS

DESCRIPTION

The MAXk is a full length PCI bus motion controller that conforms to the PCI Bus Specification, Revision 2.2. It is capable of up to 10-axis of control of which each axis can be configured as an open loop stepper, a closed loop stepper, or a servo axis.

The MAXk is powered by a PowerPC processor. This high performance processor provides a 64-bit Floating Point processor and is clocked at 266MHz. This provides the MAX with the pure processing power to update every signal of the controller, i.e. I/O bits, direction, limits, etc., at rates up to 122μs.

Every axis includes dedicated +/- over-travel limit inputs, a home input, and an auxiliary output. The home and over-travel limit inputs are TTL level inputs. The MAXk supports 16 general purpose digital I/O signals. In addition it has 4 general purpose analog inputs that can be used to sense Pressure Transducers, Dial Switches, etc. Analog inputs can also be used to control velocity override. These analog inputs have 16-bit of resolution with +/-10 VDC input. There are two general purpose analog output that use a 16-bit resolution DAC with +/- 10 VDC output.

Each axis has servo output signal capability; configured as a +/- 10V or 0-10V signal and is driven by a 16-bit DAC. The servo control loop is a PID filter with feedforward coefficients and an update rate up to 122μs. The servo output of axes not configured as a servo axis is available as a general purpose analog output. The step pulse is a TTL level, 50% duty cycle square wave that supports velocities of 0 through 4,176,00 pulses per second. Encoder feedback functionality supports quadrature encoders up to 16 MHz and is used as the servo feedback, as feedback for the stepper axes or as independent position feedback. Encoder feedback is also used to provide slip and or stall detection. SSI-Absolute encoder feedback with resolution of up to 32 bit are available for each axis upon request.

The MAXk has many user-definable parameter that can customize the controller's behavior. These parameters can be conveniently stored in Flash so that the user-defined behavior will be automatically preset at each power-up.

PROGRAMMING

MAXk motion controllers are easily programmed with ASCII character commands through an extensive command structure. These commands are combined into character strings to create sophisticated motion profiles with features of I/O and other functionality. A separate FIFO command queue for each axis is used to store the commands once they are parsed by the MAXk. The commands are executed sequentially, allowing the host to send a complex command sequence and attend to other tasks while the MAXk manages the motion process. These command queues store 2559 command values and include a command loop counter which allows multiple executions of any queued commands.

All commands are sent to the controller as two or three character ASCII strings. Some of these commands expect one or more numerical operands to follow. These commands are identified with a '#' after the command. The '#' indicates a signed integer input parameter or a signed fixed point number of the format ##.# when user units are enabled. User Units define, distances, velocity and acceleration parameters and may be inputted in inches, millimeters, revolutions, etc.
Synchronized moves may be made by entering the AA or AM command mode. This form of the command performs a context switch that allows entering commands of the format
MR#,#,#,#,#,#,#,#,#,#;
The order of axes is always X, Y, Z, T, U, V, R, S, W, K.

Numbers are entered for each axis commanded to move. An axis may be skipped by entering the comma with no parameter. The command may be prematurely terminated with a ";", i.e. a move requiring only the X and Y axes would use the command MR#,#; followed by the GO command. Each axis programmed to move will start together upon executing the GO command. The MAXk can be switched back to the single-axis mode by entering the desired single axis command such as AX.

PROGRAMMING EXAMPLES

In a typical move requirement where it is desired to home the stage then move to a specified position, the following will demonstrate the programming for a single axis:

- Initialize the velocity and acceleration parameters to a suitable value. Set a PID filter gain values. Enable the PID hold mode. Perform the home operation initializing the position counter to zero. Perform a motion to an absolute position of 10,000 and set the done flag for that axis when the move is finished.

  The following would be input from the host computer:

```
AX;
VL5000;
AC50000;
KP20;
KI1;
KD45;
CL1;
HM0;
MA10000;
GO;
ID;
```

In a move requiring a three axis coordinated move to a select position the following commands could be used:

```
AM;
VL5000,5000,5000;
AC50000,50000,50000;
MT1000,2000,3000;
GO;
ID;
```

The controller would calculate the relative velocities required to perform a straight line move from the current position to the desired absolute position so that all axes arrive at their destinations at the same time.

The following demonstrates cutting a hole with a 10,000 count radius using variable velocity contouring with circular interpolation:

- The vector velocity is set to 1000 counts per second. A contour is defined beginning at coordinates 0,0 on the X and Y axes.
- General purpose I/O7 is turned on, which could turn on the cutting torch or laser starting the cut at the center of the circle.
- A half circle is cut from the center to the outside of the hole positioning the cutting tool at the start of the hole.

The hole is then cut, the torch turned off, the stage stopped and the contour definition completed.

The following would be input from the host computer:

```
AA;
VOA0,5;
VV1000,1000;
VP0,0;
VIO0100,,0100;
VC0,5000,-180;
VC0,0,-360;
VIO0,,0100;
VV1000,0;
VP-1000,0;
VE;
```

During this sequence the VO command or an analog input may be used to vary the vector velocity from 0-200% of the program vector velocity.

SPECIFICATIONS

Velocity

0 to 4,176,000 pulses per second simultaneous on each axis

Acceleration

0 to 8,000,000 pulses per second per second

Position range

4,294,967,295 pulses (+/- 2,147,487,647)

Accuracy

Position accuracy and repeatability ±0 counts for point to point moves

Environmental

Operating temperature range: 0 to 50 degrees centigrade

Storage temperature range: -20 to 85 degrees centigrade

Humidity: 0 to 90% non-condensing

Power

+5VDC +/-5% at 1 amp typical
+3.3VDC +/-0.3% at 0.6 amps typical
+12VDC at 0.1 amp typical = +/-5%
-12VDC at 0.1 amp typical = +/-10%

Dimensions

12.283" x 4.200" x 0.475"
312 mm x 106mm x 12.06 mm

Communication Interface

Meets all signal specifications PCI Bus Specification, Revision 2.2.
Is backward compatible with MAXp.

Limit switch inputs

TTL input levels. Input sense (low or high true) selectable by command input for each axis.

Connector

Two 100-Pin SCSI type connectors for all control and I/O signals, shielded. Controller models with 5 or less axes provide only half of the possible I/Os on one connector.

Home switch inputs

TTL input levels. Input sense (low or high true) selectable by command input for each axis. Accuracy to 1 encoder count.

User definable I/O

Up to 16 bits of user definable Digital I/O. The 16 bits are user configurable that are configured as 8 inputs and 8 outputs from the factory.

Analog inputs

Four analog inputs, +/-10V, 16 bit resolution.

Analog outputs

-10V or 0 to +10V,max. 1mA each, 16 bit resolution. One per axis plus one general purpose output per 5 axes.

Step pulse output

Pulse width 50% duty cycle. Actively driven TTL level signal (max 12mA).

Direction output

Actively driven TTL level signal (max 12mA).

Encoder Feedback

Maximum 16 MHz after 4x quadrature detection. Differential TTL level signal. Single ended TTL level signal require external bias for reliable operation.

Absolute Encoders

SSI Technology
X and Y axes up to 12 bits resolution. (default)
Upon request absolute encoders up to 32 bits resolution for each axis.

Reference

PCI Bus Specification, Revision 2.2
PCB mechanical specification, IEEE 1101.1, 1101.10 and P1101.11

Software

High level expertise not required.

Over 200 ASCII character commands, expanded from current OMS command set.

Software drivers and DLLs for Windows® provided at no additional cost.

User Manual included

Servo Tuning Assistant software tools included at no additional cost.

Support software available for download at our web-site (http://www.pro-dexoms.com/)

FEATURES

PID Update Rate of 122 µs on All 10 Axes
Delivers exceptional servo control on multi-axis applications. Identical outcomes when utilizing one or all axes of motion. Configurable PID filter with feedforward coefficients.

266 MHz, 32-bit RISC Processor
Updates all signals and data points providing superior application control.

64K Shared Memory
Permits rapid data transfer to & from controller. Large size accommodates expandability to unique and custom applications.

PCI Universal Bus - 3.3 or 5.0 volts
PCI Bus Specification, Revision 2.2 compliant. Compatible with current and future PCI bus computers.

Memory
32 Mb System Memory.

Controller I/O Capabilities
4 Channels of general purpose Analog Input, with 16 bit, +/-10 VDC input
Support Quadrature Encoder Feedback up to 16 MHz.
Support for SSI Absolute encoder up to 32 bit resolution.
16 bit DAC analog resolution. Step pulses from 0 to 4,176,000 steps per second (+/- 0 steps). Backlash compensation. Custom, Parabolic, “S”-curve & Linear trajectory profiles. Real time encoder position capture. S-Curve with 4-quadrant jerk parameters.

Control signals
TTL level Digital I/O. SCSI type 100 pin connector.

Table 6-1 OUTPUT CONNECTOR PIN LISTs (J1, J2)

J1 – 100-pin connector				**J2 – 100-pin connector**			
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	X Phase +A	51	Y Phase +A	1	V Phase +A	51	R Phase +A
2	X Phase -A	52	Y Phase –A	2	V Phase -A	52	R Phase –A
3	X Phase +B	53	Y Phase +B	3	V Phase +B	53	R Phase +B
4	X Phase -B	54	Y Phase -B	4	V Phase -B	54	R Phase -B
5	X Index +	55	Y Index +	5	V Index +	55	R Index +
6	X Index -	56	Y Index -	6	V Index -	56	R Index -
7	Z Phase +A	57	GROUND	7	S Phase +A	57	GROUND
8	Z Phase -A	58	GROUND	8	S Phase -A	58	GROUND
9	Z Phase +B	59	T Phase +A	9	S Phase +B	59	W Phase +A
10	Z Phase -B	60	T Phase –A	10	S Phase -B	60	W Phase –A
11	Z Index +	61	T Phase +B	11	S Index +	61	W Phase +B
12	Z Index -	62	T Phase –B	12	S Index -	62	W Phase –B
13	U Phase +A	63	T Index +	13	K Phase +A	63	W Index +
14	U Phase -A	64	T Index -	14	K Phase -A	64	W Index -
15	U Phase +B	65	U Index +	15	K Phase +B	65	K Index +
16	U Phase -B	66	U Index -	16	K Phase -B	66	K Index -
17	GROUND	67	GROUND	17	GROUND	67	GROUND
18	X Positive Limit	68	X Negative Limit	18	V Positive Limit	68	V Negative Limit
19	Y Positive Limit	69	Y Negative Limit	19	R Positive Limit	69	R Negative Limit
20	Z Positive Limit	70	Z Negative Limit	20	S Positive Limit	70	S Negative Limit
21	T Positive Limit	71	T Negative Limit	21	W Positive Limit	71	W Negative Limit
22	U Positive Limit	72	U Negative Limit	22	K Positive Limit	72	K Negative Limit
23	GROUND	73	GROUND	23	GROUND	73	GROUND
24	X Home	74	X Direction	24	V Home	74	V Direction
25	Y Home	75	Y Direction	25	R Home	75	R Direction
26	Z Home	76	Z Direction	26	S Home	76	S Direction
27	GROUND	77	GROUND	27	GROUND	77	GROUND
28	T Home	78	T Direction	28	W Home	78	W Direction
29	U Home	79	U Direction	29	K Home	79	K Direction
30	GROUND	80	GROUND	30	GROUND	80	GROUND
31	X Aux	81	X Step	31	V Aux	81	V Step
32	Y Aux	82	Y Step	32	R Aux	82	R Step
33	Z Aux	83	Z Step	33	S Aux	83	S Step
34	**5 Volts**	84	GROUND	34	**5 Volts**	84	GROUND
35	T Aux	85	T Step	35	W Aux	85	W Step
36	U Aux	86	U Step	36	K Aux	86	K Step
37	GROUND	87	GROUND	37	GROUND	87	GROUND
38	IO0	88	IO1	38	IO8	88	IO9
39	IO2	89	IO3	39	IO10	89	IO11
40	IO4	90	IO5	40	IO12	90	IO13
41	IO6	91	IO7	41	IO14	91	IO15
42	GROUND	92	GROUND	42	GROUND	92	GROUND
43	ADC 0	93	ADC 1	43	ADC 2	93	ADC 3
44	GROUND	94	GROUND	44	GROUND	94	GROUND
45	X Servo	95	Y Servo	45	V Servo	95	R Servo
46	GROUND	96	GROUND	46	GROUND	96	GROUND
47	Z Servo	97	T Servo	47	S Servo	97	W Servo
48	GROUND	98	GROUND	48	GROUND	98	GROUND
49	U Servo	99	DAC 0	49	K Servo	99	DAC 1
50	GROUND	100	GROUND	50	GROUND	100	GROUND

Table 6-2

IOMAXnet – Terminal Block Pin-Out

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
52	Y Phase –A	51	Y Phase +A	2	X Phase -A	1	X Phase +A
54	Y Phase -B	53	Y Phase +B	4	X Phase -B	3	X Phase +B
56	Y Index -	55	Y Index +	6	X Index -	5	X Index +
58	GROUND	57	GROUND	8	Z Phase –A	7	Z Phase +A
60	T Phase –A	59	T Phase +A	10	Z Phase –B	9	Z Phase +B
62	T Phase –B	61	T Phase +B	12	Z Index -	11	Z Index +
64	T Index -	63	T Index +	14	U Phase –A	13	U Phase +A
66	U Index -	65	U Index +	16	U Phase –B	15	U Phase +B
68	X Negative Limit	67	GROUND	18	X Positive Limit	17	GROUND
70	Z Negative Limit	69	Y Negative Limit	20	Z Positive Limit	19	Y Positive Limit
72	U Negative Limit	71	T Negative Limit	22	U Positive Limit	21	T Positive Limit
74	X Direction	73	GROUND	24	X Home	23	GROUND
76	Z Direction	75	Y Direction	26	Z Home	25	Y Home
78	T Direction	77	GROUND	28	T Home	27	GROUND
80	GROUND	79	U Direction	30	GROUND	29	U Home
82	Y Step	81	X Step	32	Y Aux	31	X Aux
84	GROUND	83	Z Step	34	5 Volts	33	Z Aux
86	U Step	85	T Step	36	U Aux	35	T Aux
88	IO1	87	GROUND	38	IO0	37	GROUND
90	IO5	89	IO3	40	IO4	39	IO2
92	GROUND	91	IO7	42	GROUND	41	IO6
94	GROUND	93	ADC 1	44	GROUND	43	ADC 0
96	GROUND	95	Y Servo	46	GROUND	45	X Servo
98	GROUND	97	T Servo	48	GROUND	47	Z Servo
100	GROUND	99	DAC 0	50	GROUND	49	U Servo

ORDERING INFORMATION

Model	Computer Interface	Axes	Servo / Stepper	I/O					
				Digital				Analog	
				Limit	Auxiliary	Home	General Purpose	In	Out
MAXk-1000	Universal PCI	1	User Definable	2	1	1	8	2	2
MAXk-2000		2	User Definable	4	2	2	6	2	3
MAXk-3000		3	User Definable	6	3	3	8	2	4
MAXk-4000		4	User Definable	8	4	4	8	2	5
MAXk-5000		5	User Definable	10	5	5	8	2	6
MAXk-6000		6	User Definable	12	6	6	16	4	8
MAXk-7000		7	User Definable	14	7	7	16	4	9
MAXk-8000		8	User Definable	16	8	8	16	4	10
MAXk-9000		9	User Definable	18	9	9	16	4	11
MAXk-A000		10	User Definable	20	10	10	16	4	12
CBL58-3M	100-Pin, 12 ft. cable (one per 5 axes)								
IOMAXnet	100-Pin Connector Breakout Module (one per 5 axes)								

INDEX