An Introductory Guide to Motion Control Programming...

Precision MicroControl (PMC) motion controllers are successfully deployed in a wide variety of applications such as electronic assembly, electro-optics and semiconductor machine automation. PMC recognizes that no matter what the application, good software tools are critical to the success of any automation project.

To speed system development, PMC offers a comprehensive and powerful high-level Motion Control Application Programming Interface (MCAPI), as well as our Motion Integrator™ suite of graphical setup, tuning and diagnostic utilities. These software tools are designed to help you get up-and-running quickly – whether your control program requires just a few simple motion commands, or the flexibility and power of a multi-threaded C++ host application.

To further simplify system integration and eliminate any hidden costs, all software is included with PMC motion control cards at no extra charge. And to ensure that you get the most out of your programming investment, all functions and commands are compatible across the entire PMC product family.

- Fully programmable in C/C++, Delphi & Visual Basic
- Drivers for Windows 98/2000/NT/XP and Linux
- Simple command-based programming also available
- Includes Motion Integrator™ setup & diagnostic tool suite
-Compatible across PMC controller family
For the experienced programmer, the MCAPI provides a standard set of functions that can be called from application programs written in C/C++ and other high level languages such as Visual Basic and Delphi (Pascal). In addition, the Motion VI Library provides the LabVIEW and BridgeVIEW programmer with graphical access to the MCAPI. For easy integration with Microsoft COM enabled software (Component Object Model supporting ActiveX components), a standard COM interface to the MCAPI is also provided.

The MCAPI is fully supported under Windows 98/2000/NT/XP and Linux. For the Windows and Linux developer, the MCAPI is a Dynamic link Library (DLL) that communicates with the control card through 32-bit installable device drivers.

The functions of the MCAPI have names consistent with the operations they perform, making programs easy to read and practically self-documenting. For example, the function call to move 1 axis to position 1000 would appear in a C program as:

```
MCMoveAbsolute( ctrlr,1,1000,0);
```

The function call to read the position of axis #8 into a variable is:

```
Position = MCGetPosition (ctrlr,8);
```

The Motion Control API (MCAPI) was designed from the ground up to include the power and flexibility that experienced C-programmers have come to expect. The design of the MCAPI includes a high degree of backwards- and cross-platform compatibility. Code written four, five, or even six years ago will run on the latest versions of the MCAPI.

For high-performance motion control applications, communication speed is critical. The MCAPI uses the motion control card’s dual ported memory and high-speed direct binary interface for optimum throughput.

**Partial Listing of MCAPI Functions**

- **MCEnableDigitalIO( cttrr, channel, state );** Change output state of digital I/O channel
- **MCSetJogConfig( cttrr, axis, lpjog );** Set jog configuration
- **MCEnableGearin( cttrr, axis, masis, ratio, state );** Configure and enable / disable gearing
- **MCDirection( cttrr, axis, dir );** Set direction of travel
- **MCEnableAxis( cttrr, axis, state );** Turn axis on or off
- **MCGetPosition( cttrr, axis );** Get current position
- **MCGetStatus( cttrr, axis );** Get status value from axis
- **MCGetVelocity( cttrr, axis );** Get velocity value
- **MCGo( cttrr, axis );** Start motion
- **MCGoHome( cttrr, axis );** Start axis or axes moving to home position
- **MCLearnPosition( cttrr, axis, index );** Store current position to point memory
- **MCMoveAbsolute( cttrr, axis, index );** Move to an absolute position
- **MCMoveRelative( cttrr, axis, distance );** Move a relative distance from current position
- **MCSetAcceleration( cttrr, axis, rate );** Set acceleration value
- **MCSetDeceleration( cttrr, axis, rate );** Set deceleration value
- **MCSetServoOutputPhase( cttrr, axis, select );** Set servo output phase
- **MCGetScale( cttrr, axis, scale_factors );** Get current scale factors
- **MCSetScale( cttrr, axis, scale_factors );** Set scale factors for unit conversion
- **MCSetFilterConfig( cttrr, axis, lpflf );** Set PID filter values

A representative sample of the more than 130 MCAPI functions. Contact PMC for an up-to-date list. We continually add functions to support new features.
Motion Control Programming Options

WinControl is a powerful yet easy to use Windows based terminal emulator for MCCL programming. MCCL commands can be entered from the keyboard or downloaded from an ASCII text file.

The Windows MCAPI on-line help includes links to programming examples and associated commands.

Software is Compatible Across the Entire Family of PMC Controllers

Tight integration with popular programming environments includes syntax coloring to allow easy identification of MCAPI specific code.

WinControl is a powerful yet easy to use Windows based terminal emulator for MCCL programming. MCCL commands can be entered from the keyboard or downloaded from an ASCII text file.
MCCL – Motion Control Command Language

- Intuitive, easy-to-use commands
- Store & execute multiple programs on-board
- Solve any application, from basic to the most complex
- Ideal for prototyping and embedded control applications

PMC’s motion control cards can execute more than 175 MCCL commands, allowing you to perform a wide variety of tasks with a simple on-board command language. Setting a motor’s maximum speed, moving a motor to a specific position, or even reporting the current position are just some of the operations that can be performed using the MCCL commands.

Each MCCL command can easily be identified by a two-letter mnemonic. The letters are easy to remember because they relate to the function the command performs. The format for all commands is the same as the example below:

```
1 MA 10.5
```

This command causes axis 1 to Move Absolute to position 10.5. By placing commas (,) between multiple commands, they can all be issued at the same time to initiate synchronized multi-axis motion.

The MCCL language includes commands for conditional execution branching and looping. Using these commands, complex control operations can be implemented in user-written “macro” routines. Multiple commands can be linked, permanently stored in the card’s memory as a macro command, and used at any time. Macro commands can be written to perform any motion, from a simple homing routine, to controlling an entire machine without the intervention of a host computer.

As with the Windows version of the MCAPI, the MCCL commands can be sent to the card via three different interfaces:

- PC bus (ISA or PCI using the terminal emulator software utilities)
- RS-232/422 serial port
- IEEE-488 interface

With a terminal emulator utility running on the host PC, typing on the keyboard transfers one character at a time to the motion control card. Any response from the card will be displayed on the host computer screen. Motion control commands can also be placed in an ASCII text file and downloaded to the card.

### Partial Listing of MCCL Command Set

<table>
<thead>
<tr>
<th>Setup Commands</th>
<th>Motion Commands</th>
<th>Reporting Commands</th>
<th>Macro Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP Profile Parabolic</td>
<td>CM Contour Mode</td>
<td>AF Tell Pos. Aux. Encoder</td>
<td>ET Escape Task</td>
</tr>
<tr>
<td>PS Profile S curve</td>
<td>CP Contour Path</td>
<td>TF Tell Following Error</td>
<td>GT Generate Task</td>
</tr>
<tr>
<td>SA Set Acceleration</td>
<td>CR Arc Center Relative</td>
<td>TO Tell Optimal Position</td>
<td>MC Macro Call</td>
</tr>
<tr>
<td>SD Set Derivative Gain</td>
<td>FE Find Edge</td>
<td>TP Tell current Position</td>
<td>MD Define as Macro</td>
</tr>
<tr>
<td>SE Stop on Follow Error</td>
<td>FI Find Index</td>
<td>TS Tell Servo Status</td>
<td>MJ Macro Jump</td>
</tr>
<tr>
<td>SG Set Prop. Gain</td>
<td>GH Go Home</td>
<td>TT Tell Target Position</td>
<td>RM Reset (clear) Macros</td>
</tr>
<tr>
<td>SH Step Half/Micro</td>
<td>GO Start in Velocity Mode</td>
<td>TX Tell Cont. Count</td>
<td>TM Tell Macro</td>
</tr>
<tr>
<td>SI Set Int. Gain</td>
<td>MA Move Absolute</td>
<td>TZ Tell Index Position</td>
<td></td>
</tr>
<tr>
<td>SQ Set Torque</td>
<td>MF Motor Off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS Set Slave ratio</td>
<td>MN Motor On</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SV Set Velocity</td>
<td>MR Move Relative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US User Scale</td>
<td>PM Position Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA Vector Acceleration</td>
<td>QM Torque Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VD Vector Deceleration</td>
<td>SM Set Master</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VG Velocity Gain</td>
<td>SN Synchronization On</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VO Velocity Override</td>
<td>ST Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VV Vector Velocity</td>
<td>VM Velocity Mode</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A representative sample of the more than 175 MCCL commands.
Motion Control Programming Options

High Level Languages
- C
- C++
- Visual Basic
- Pascal

Advanced Development Environments
- Delphi
- Lab Windows
- Visual C/C++

Visual Programming
- Visual Basic
- LabVIEW
- BridgeVIEW

Drivers
- OLE controls
- LabView VI

MCCL ASCII Command Interface
(Win Control Terminal Emulator)

Motion Control API (MCAPIMCCL)

Low-Level Device Driver / DLL

PMC Motion Control Card

Motion control applications can run on the host PC, the PMC controller, or both.

- **Host-based application programs:**
  For sophisticated applications running on a host PC, the Motion Control API (MCAPI) provides seamless integration between PMC motion control systems and popular high-level programming languages.

- **Embedded application programs:**
  For applications and routines running on-board the PMC controller, the Motion Control Command Language (MCCL) is easy-to-use and frees the host PC for other tasks.

Software included with each controller includes many high-level programming examples and pre-built Windows dialog boxes with source code, as well as extensive online help and PMC’s Motion Integrator™, a comprehensive suite of setup, tuning and diagnostic tools.

Get your controller installed and running in hours instead of days with PMC’s graphical and intuitive Motion Integrator™ setup, tuning & diagnostic programs.

Two of the many high-level programming examples - available with fully annotated source code

A Servo Tuning and a System Setup dialog box

Motion Dialog boxes are pre-built for all common setup tasks, so you can incorporate them into your application program with a single function call. They are already written, so you don’t have to.

A complete VI library with detailed motion control icons is provided for LabVIEW programmers

PMC’s Motion Control API for Windows 95/98/NT/2000 includes complete function libraries for:

- Visual C/C++
- Visual Basic
- Borland Delphi
- LabVIEW
- BridgeVIEW
- Watcom C/C++

PMC continuously develops new software tools and provides custom software solutions to qualified OEMs. Contact a PMC application engineer to discuss your requirements.
Motion Control Programming Examples

PMC provides programming support for DOS, Windows, and stand-alone applications. Below are two motion control programming examples. Each example shows how to program the same motion using MCCL commands and MCAPI functions.

Basic Positioning

Any or all axes can be easily programmed for independent motion with a minimum of commands.

Move axis 1, wait for 5 msec, return to starting position

MCCL
1SV15,1SA5, 1DS5 ; Set Axis #1 vel., Accel. & Decel
1MR3.5,1WS0.005,1MR-3.5 ; Move axis #1, 3.5", wait 5 msec,
MCCL using “C”
MCSetVelocity( 1, 15.0 ); // set axis 1 velocity
MCSetAcceleration( 1, 5.0 ); // set axis 1 acceleration
MCSetDeceleration( 1, 5.0 ); // set axis 1 deceleration
MCWaitForStop( 1, 0.005 ); // wait 5 msec after axis 1 stops
MCMoveRelative( 1, -3.5 ); // move axis 1 by -3.5 inches

MCCL using “basic”
Call MCSetVelocity( 1, 15.0 ); // set axis 1 velocity
Call MCSetAcceleration( 1, 5.0 ); // set axis 1 acceleration
Call MCSetDeceleration( 1, 5.0 ); // set axis 1 deceleration
Call MCMoveRelative( 1, 3.5 ); // move axis 1 by 3.5 inches
Call MCWaitForStop( 1, 0.005 ); // wait 5 msec after axis 1 stops
Call MCMoveRelative( 1, -3.5 ); // move axis 1 by -3.5 inches

Linear and Circular Contouring

PMC’s motion control cards support simultaneous Linear and/or Circular Contouring. Any number of axes (as many as eight) with any combination of Servo and Stepper motors can be used. The on-board CPU will compute points on the path at intervals of 125 µsec, while still allowing on-the-fly changes of feed rate, acceleration, deceleration, and PID parameters. For example:

3D Linear Interpolated move

MCCL
1VV10,1VA50,1VD50
1CM1,2CM1,3CM1
1CP1,1MA15,2MA15,3MA5

MCCL using “C”
MCSetVectorVelocity = 10.0 ' set contour velocity
MCSetVectorAccel = 50.0 ' set contour acceleration
MCSetVectorDecel = 50.0 ' set contour deceleration
Call MCSetContourConfig( 1, Contour) ' initialize axis with contour
Call MCSetOperatingMode( 1, 1, CONTOUR ) ' axis one is in control
Call MCSetOperatingMode( 2, 1, CONTOUR ) ' axis two linked to axis one motion
Call MCSetOperatingMode( 3, 1, CONTOUR ) ' axis three linked to axis two
Call MCContourPath( 1, LINEAR, " 1MA15,2MA15,3MA5" )

Motion Control Integration and Diagnostic Tools

To assist the machine builder, powerful software tools are included with the Windows MCAPI. For the programmer, there is a wealth of commented sample source code. And our Motion Integrator™ suite of setup, tuning and diagnostic programs will help you get your system up and running in no time.

Motion Control Programming Examples

We continuously develop new software tools. Please consult our factory for the latest available software.