OFFICIAL

PROCEEDINGS

OF THE THIRTEENTH

MOTOR-CON '88

INTERNATIONAL

CONFERENCE

OCTOBER 3-6, 1988
DEARBORN, MICHIGAN

OFFICIAL

PROCEEDINGS

OF THE THIRTEENTH

INTERNATIONAL

MOTOR-CON '88

CONFERENCE

OCTOBER 3-6, 1988
DEARBORN, MICHIGAN

This Book the Property of

Copyright © 1988 by
Intertec Communications, Inc.

Library of Congress Catalog Card Number 88-082481

ISBN 0-931033-12-8

Thirteenth International Motor-Con Conference October 1988

Published by Intertec Communications, Inc.
2472 Eastman Avenue, Bldgs. 33-34
Ventura, California 93003-5774, USA

MOTOR-CON '88 - OCTOBER

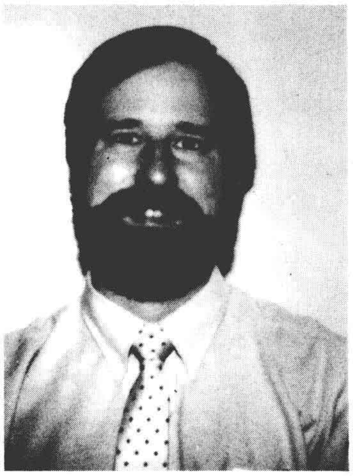
TABLE OF CONTENTS

Any papers received after publication date are not included in these Proceedings and may be obtained by writing directly to the authors. Papers received after deadline date (but before publication) will be found under Late Listings, and at the back of this book.

An Advanced Motion Control Development System for Custom Applications

By

David T. Robinson, Marketing Manager
and
David E. Halpert, Director of Technology
Creonics Inc. Lebanon, New Hampshire USA



Abstract: An integrated programming, debugging, and operating environment for developing custom high-performance motion control applications is presented. This new system architecture integrates all phases of software development and support making custom motion control systems fast and easy. Included are discussions of the IBM PC-resident software development and debugging program, the target motion controller including its resident operating system, and the advanced motion control language itself.

INTRODUCTION

One of the largest problems facing the user of motion control equipment today is programming. No matter what the application, some level of programming is required to take a general-purpose digital motion controller and adapt it to the requirements of the specific application. In some well-served market niches, special-purpose dedicated controllers such as CNCs for machine tools, feeders for roll-feeding applications, etc. have appeared. In many cases, these dedicated controllers are adequate for the application. They have some flexibility to handle variations in basic machine operation, but usually this is very narrow.

There is a large number of motion control applications that do not fall into one of these rigidly defined niches. There are many OEM machine manufacturers that have a specialized control problem which is unique to their particular process. Similarly, there are a great many End Users or Systems Integrators who are building one-of-a-kind production machines or process lines which have complex motion control aspects and/or associated customized process logic.

In each of these cases special programming has to be done -- usually by the control system manufacturer and at the customer's expense -- to implement the desired feature in the dedicated controller. It is often an expensive and time consuming process to give sufficient specifications to the control manufacturer for him to do the software. Also, many process problems are complex and require a special knowledge of the application. The customer is then stuck with a control that's not quite standard, and with a software solution to a problem which may not even be well understood. This make support very difficult from both the control system manufacturer and the customer's viewpoint.

To answer the need for an easily programmable motion controller for custom applications, a user-friendly *applications-oriented* programming language is required. Such a language must allow the application or process expert (rather than a computer programmer) to write the application-specific program.

In addition, a complete and easy-to-use application development system is required so that the custom controls incorporating this language are easy to support. This is not only for the benefit of the ultimate end-user, but also for the system integrater or OEM who develops the custom control system, and the motion control manufacturer. If the language and its associated application development system are easy to use, the system integrater or OEM is well prepared to react quickly to the inevitable specification changes which accompany any automation job.

PROGRAMMING LEVELS

In any custom motion control application, there are three levels of programming which are required. As shown in Figure 1, the top level is the application program itself -- the software which gives the general purpose motion controller its custom "personality" for the specific application. The application program is usually written by an Applications Engineer familiar with the requirements of the machine or process, but not necessarily a computer programmer.

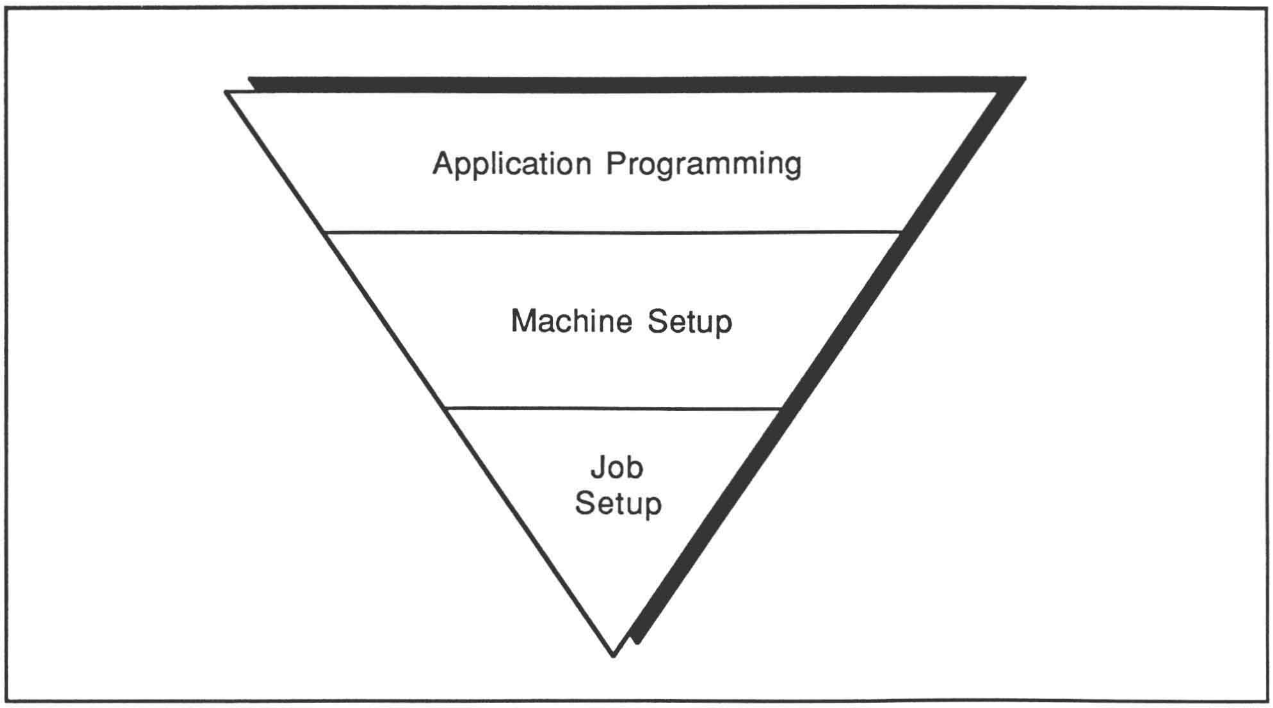


Figure 1
Custom Motion Control Programming Levels

Once the application program is written and de-bugged, and the control system integrated with the machine or process, the machine-specific setup parameters required by the application must be programmed. This second Machine Setup level of programming includes such things as setting up the drive system and tuning the servo gains, programming the feedback conversion constants, homing and overtravel configuration, etc. This level of programming is usually done once at machine installation by the installation technician.

The third level of programming is done by the machine operator as the machine is used in actual production. This third Job Setup programming level allows the machine operator to change process parameter values to alter the "recipe" as required. This is usually accomplished via some type of operator interface connected directly to the motion controller on the machine. For example, in a sheet feeding application, the operator may need to be able to program the desired Feed Length, Feed Rate, Number of Sheets, etc., as required by the day's production schedule.

The motion control language and its associated application development system must address all three levels of programming. Ideally, they should allow the applications programmer to not only write the application program itself, but also to tailor the operator interface for the specific application. The Machine Setup programming should be menu-driven so that the installation technician can set up the motion control system quickly and easily.

THE ELEMENTS OF A CUSTOM MOTION CONTROL SYSTEM

The elements of a custom motion control system are shown in Figure 2, and correspond to the three programming levels discussed above.

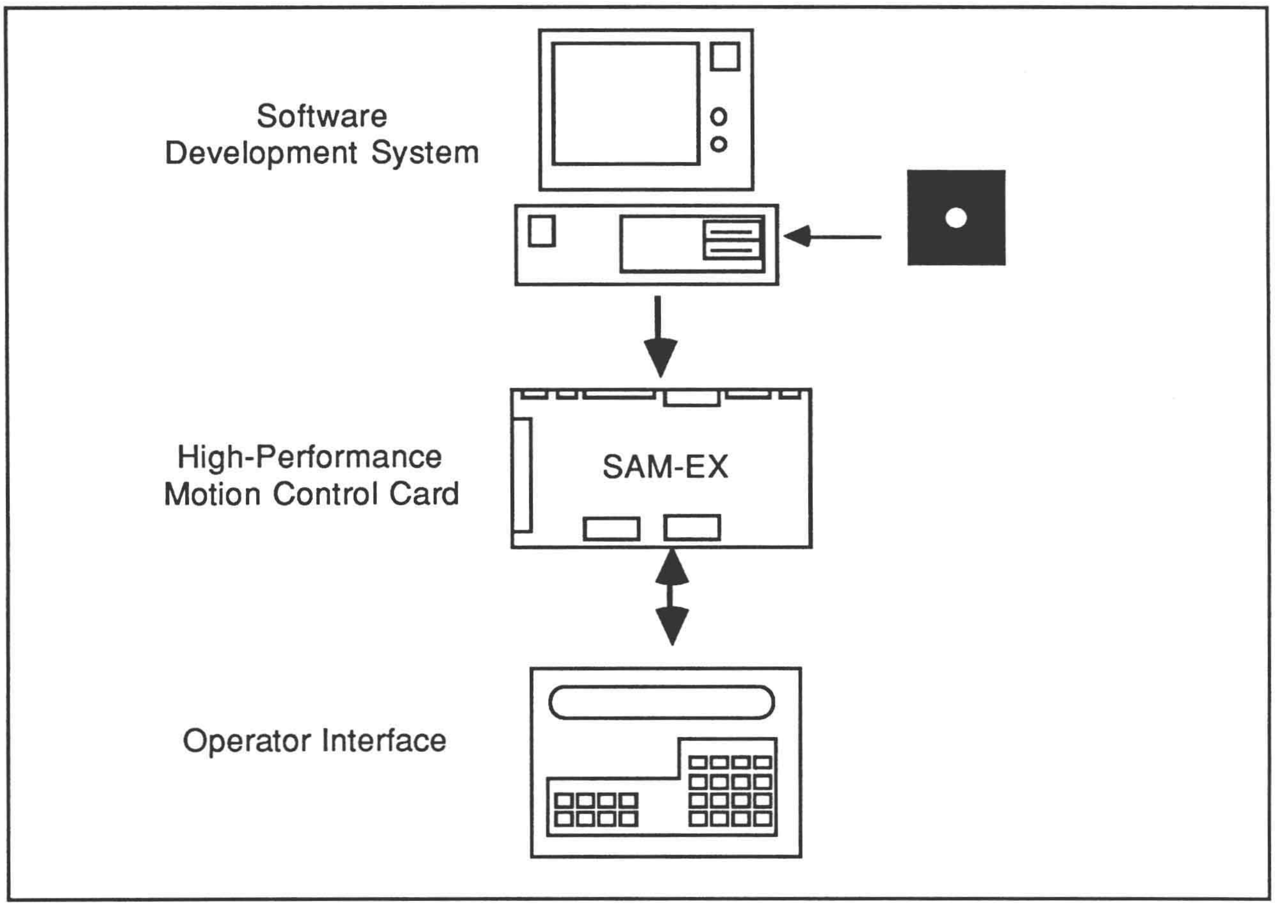


Figure 2
Elements of a Custom Motion Control System

Rather than incorporate the software development system into the motion controller as is done with some general purpose motion controllers, an IBM PC or compatible personal computer is used for application software development. By harnessing the power of the PC via a dedicated motion control development system software package, many sophisticated programming tools not available in a dedicated motion controller may be incorporated, making software development as easy as possible. This also simplifies the operating system software in the motion controller itself by eliminating the software development functions.

The high-performance Motion Control Card (MCC) executes the application program to control the machine in the desired fashion and provides the menu-driven Machine Setup programming level. It also closes and stabilizes the servo loop(s), handles the discrete I/O, and communicates with the operator interface. The Creonics SAM-EX is a typical high-performance Motion Control Card.

The exact form of the operator interface depends on the specific application. In a completely automated environment, a central host computer or PLC may download the required values at the request of the operator. In stand-alone applications, an operator control station with an alpha-numeric display and keypad or a control panel with thumbwheels and pushbuttons may be used. The MCC should be capable of interfacing to all of these different operator interfaces.

AN APPLICATIONS-ORIENTED LANGUAGE FOR CUSTOM MOTION CONTROL

In response to the above needs, Creonics has developed ***ACCEL***, an Advanced Custom Control Engineering Language. ***ACCEL*** provides an applications-oriented programming language which allows application experts to provide custom motion control solutions using general purpose motion control hardware.

ACCEL is a powerful yet simple to use motion and process control language which allows a complete custom "personality" to be stored and executed within the high-performance Motion Control Card. The MCC, once loaded with the ***ACCEL*** program, becomes a dedicated, stand-alone machine controller capable of many sophisticated motion and process control functions.

There are three essential elements which are keys to ***ACCEL***'s customization capability. The first is the ability to handle variables and evaluate mathematical expressions on-the-fly. The second element is a simple set of commands which allow complete customization of the operator interface. The third element is a complete set of built-in high-level motion functions to allow complex motion requirements to be easily programmed.

Variables and Expressions

Variables come in two types: User Variables and Internal Variables. User Variables are defined by the application programmer and can be used for whatever function is required. Internal Variables allow motion-related values such as current axis position, current velocity, etc. to be used directly in the application program. Expressions allow addition, subtraction, multiplication, and division operations to be performed on variables and constants, providing real-time calculation capability.

User Variables have a definable numeric format, optional range checking to reject illegal values and an optional prompt string to be used when a value for that variable is entered by the operator via the operator interface. For example, consider the simple material feeding application shown in Figure 3.

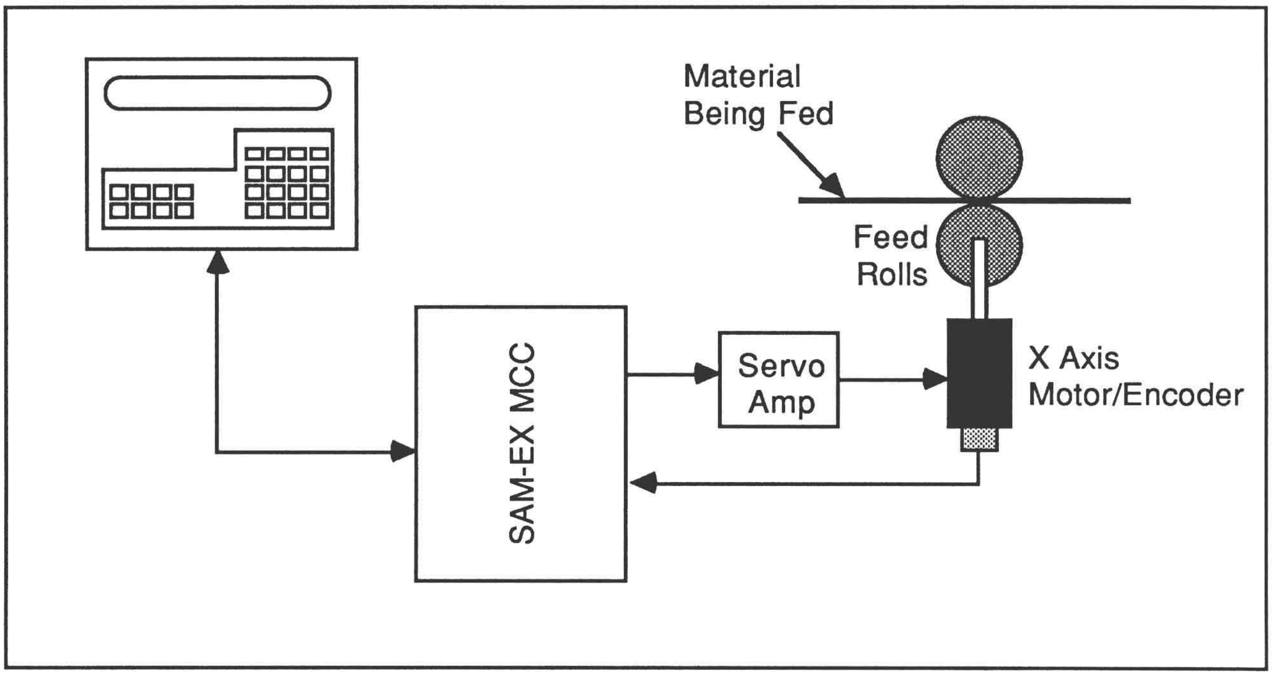


Figure 3
Simple Material Feeder Application

User Variable V0 might be used for the Feed Length and V1 for the Feed Rate. In this case, each variable is defined using a DEF statement which specifies the numeric format (F=), the upper and lower value limits (U= and L=), and the prompt string (P=) as shown below:

```
DEF V0 [F=XX.XXX, U=50.000, P="Feed Length (inches):"]
DEF V1 [F=XX, L=10, P="Feed Rate (% of Maximum):"]
```

In this application, the actual feed is an incremental move with a distance of V0 at a rate of V1. This can be accomplished as shown below:

```
MIV0,@V1
```

Programming a Custom Operator Interface

The operator interface for Job Setup programming (see Figure 1) is fully programmable. ***ACCEL*** allows the application programmer to determine what parameters the machine operator is allowed to enter. In many languages available for controls, it requires a significant programming effort to develop the software associated with an operator interface. ***ACCEL*** is specifically designed with a built-in preconfigured operator interface which is suitable for wide range of operator parameter entry. It incorporates a prewritten menu structure and easily configured automatically updating status display. In the code fragment given in the previous paragraph, the two definition (DEF) statements used to describe the variables used for operator specification of the process parameters for the example feeder application also include customized specifications for the necessary operator prompts and range limits. No additional program statements need be included to handle the operator interface.

In this same application, a complete definition of an auto refreshing status display including 2 fields of numeric data for the current feeder position and speed can be defined by the following single ***ACCEL*** statement:

```
DISP ["Feed Pos"=VPX," Speed"=VUX]
```

For more specialized operator interface requirements, language statements are available to easily incorporate custom menus, English-language prompts, direct numeric parameter editing with range-checking, and toggle-selected options into the operator interface. In this way, a more sophisticated and unique operator interface -- customized for the specific application -- can be easily programmed with a minimal number of statements.

In the simple feeder example discussed above, the values for User Variables V0 and V1 are entered via a menu selection included as part of ***ACCEL***'s built-in Standard Operator Interface. However, if the application programmer desires to prompt the machine operator for these values when they are required (rather than having them be set in beforehand), this can be implemented as shown below:

```
?"Feed Length (Inches)",V0
?"Feed Rate (% of Maximum)",V1
```

When these commands are encountered in the ***ACCEL*** program, the prompt message ("Feed Length (inches)" or "Feed Rate (% of Maximum)") is displayed and the operator may then enter the appropriate value.

In many applications, different operating modes such as Manual, Semi-Automatic, and Automatic must be selected by the operator. The most straightforward way of doing this is to display the three options sequentially and allow the operator to select the appropriate one. This type of selection is called "Toggling" and can be implemented directly using a single ***ACCEL*** statement:

```
?T"Operating Mode?",V10,"MANUAL","SEMI-AUTO","AUTO"
```

When this statement is encountered in the application program, the currently selected mode (as determined by the current value of V10) would be displayed.

```
Operating Mode? MANUAL
```

The machine operator may then select a new mode by pressing the TOGGLE key until the desired mode appears and confirm the new mode by pressing ENTER. The value of the selected mode (1, 2, or 3) is then stored as the value of V10. V10 can then be used within the application program to select the appropriate action.

In more complex applications, a more sophisticated menu structure may be required for the operator interface. Such custom menus can be implemented within the application program using several of the built-in high-level functions. The following program fragment implements a simple single-level menu which might be used in a drill and tap application.

```
P"|DRILL|TAP|RETRACT|PAUSE|"          \Display Menu
$1                                    \Loop-back Label
?K1                                   \Wait for a key
(VK=49)>10                            \If Key 1, do Drilling
(VK=50)>20                            \If Key 2, do Tapping
(VK=51)>30                            \If Key 3, Retract
(VK=52)>40                            \If Key 4, Pause
>1                                    \Illegal key - loop back
```

More sophisticated menus with sub-menus can be implemented in a similar fashion.

High-Level Motion Functions

Many application problems require additional motion control functions in addition to the usual incremental and absolute moves. By identifying fundamental motion building blocks and implementing at a high level so the the application programming does not need to be concerned with the internal details, a wide range of sophisticated customized motion control operations can be easily implemented. Examples of such functions include Software Limit Switches and Position Registration.

Software Limit Switches allow programming certain actions (like changing axis speed) to occur at pre-defined axis positions, much the same as hard-wired electrical limit switches. The following ***ACCEL*** command sets up a software limit switch to slow the X Axis to a creep speed when it is 1 inch away from its programmed destination. The creep speed is stored in User Variable V12 and the programmed destination in V40.

```
WX(V40-1),>@V12
```

Position Registration is a common requirement whenever there is slippage in the mechanical transmission (belts, feed rolls, etc.), or whenever axis position must be calibrated to some physical event. Printing, packaging, and roll feeders are examples of applications requiring position registration.

For example, consider a roll-feed application where the position of the material being fed must be re-calibrated to a lead-edge sensor to compensate for slippage in the feed rolls. Such an application is illustrated in Figure 4.

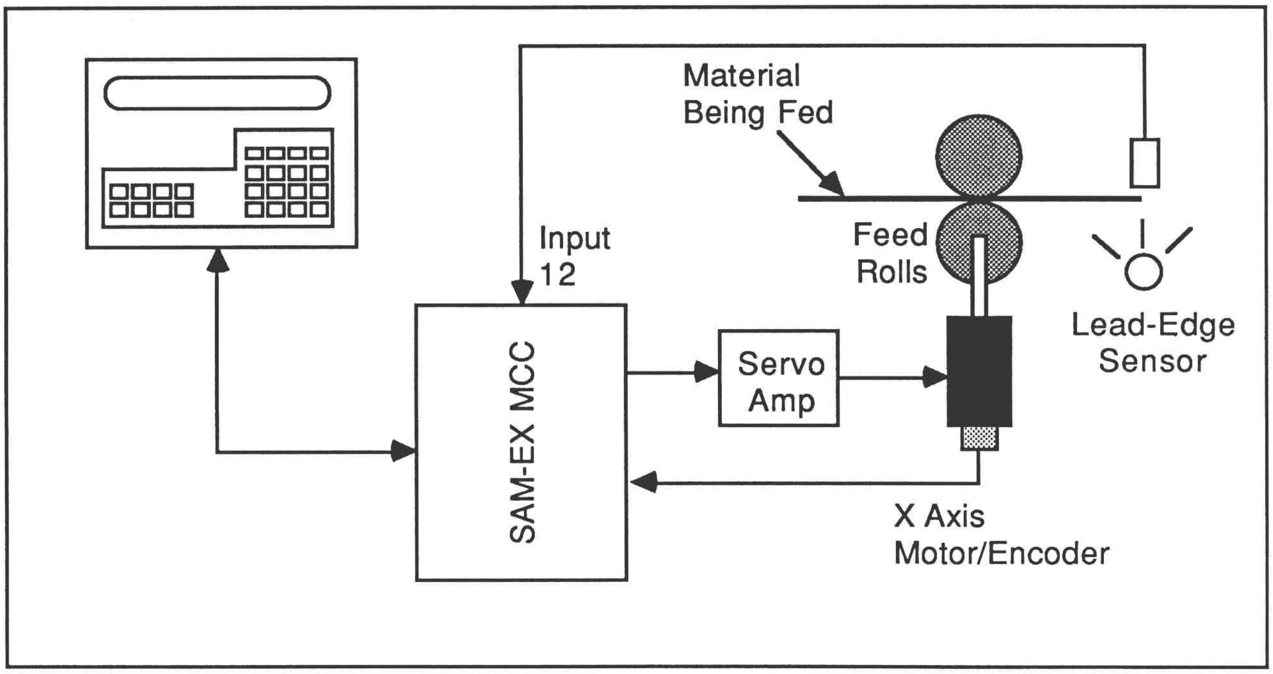


Figure 4
Position Registration in a Roll-Feeding Application

If the lead-edge sensor is connected to discrete input 12 on the MCC, the following command registers the material to the sensor by resetting the axis position to zero when the lead-edge sensor is activated.

```
RX12,1,N0.00
```

A non-zero position may also be specified to compensate for any offset between the sensor and the desired zero position of the material.

These high-level motion functions are only truly useful in the real-world if they can be executed in real time without regard for the position error resulting from the execution time of the statement itself. ***ACCEL*** is designed such that the all time-critical commands are executed as fast as if it were written in assembly code. This powerful feature gives the application programmer the speed benefits of assembly-language with the ease of programming of a high-level language.

MAKING APPLICATION PROGRAMMING EASY

To allow the application program development to proceed in parallel with the machine construction, ***ACCEL*** allows the application program to be developed "off-line". In other words, the actual motion controller itself is not be required to develop the application program. The application development system, called ***ACCEL*** Programmer's Workshop (APW), is a software package which runs on an IBM PC and provides the applications engineer or system integrater with a complete set of tools necessary for developing custom application programs using ***ACCEL***.

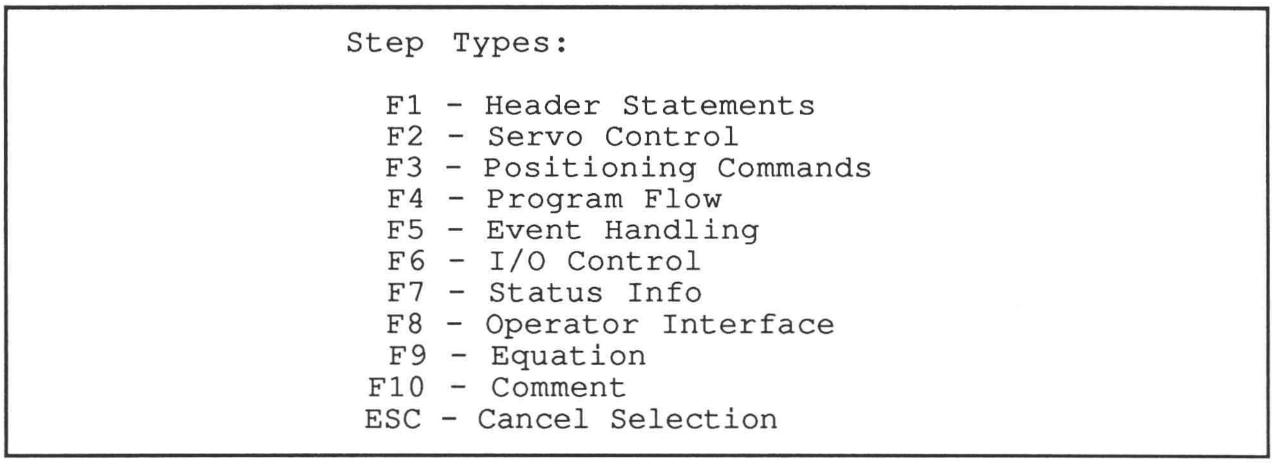
Prompted Programming

In any programming language, one of the most difficult things for a non-programmer to master is the command syntax -- the rules by which commands are constructed. Not only must all the commands be memorized, but also a strict set of rules as to how each command is constructed. This can be a daunting proposition in any language, but especially in a specialized but highly versatile language such as ***ACCEL***. Once the command construction and statement syntax are mastered, however, application programs can be written quickly and efficiently using any standard text editing program.

To allow both the novice and the experienced ***ACCEL*** programmer to create application programs easily, APW incorporates a full-featured text editor with a built-in prompted English-language command constructor accessible via pop-up menus. Available at any time during program development, this command constructor, called EasyACCEL, actually writes syntactically correct ***ACCEL*** commands by prompting the programmer for the desired action at every step of the way.

EasyACCEL provides a number of important benefits to the application programmer. Since its use is optional, an experienced programmer is not handicapped and can always write program statements directly. The novice, however, has a powerful tool with which to learn the new language and can begin programming immediately with very little reference to the manual. In addition, since the actual program statements are constructed on-screen as menu choices are made, EasyACCEL is an excellent on-line reference and learning tool for correct command construction and syntax.

Accessing EasyACCEL while in the text editor requires a single keypress. A menu of command type options (Figure 5) is then displayed, and the desired selection made. Another sub-menu of options specific to the selected command type is then displayed and a selection made. This process continues until all selections relating to that command are made. As each selection is made, the correct command syntax for that selection is displayed on-screen at the appropriate point in the program. When all selections relating to the command have been made, the command is complete and inserted into the program.



```
Step Types:

  F1 - Header Statements
  F2 - Servo Control
  F3 - Positioning Commands
  F4 - Program Flow
  F5 - Event Handling
  F6 - I/O Control
  F7 - Status Info
  F8 - Operator Interface
  F9 - Equation
 F10 - Comment
 ESC - Cancel Selection
```

Figure 5
EasyACCEL Step Type Menu

For example, suppose the application programmer needs to move the axis a certain distance at a certain speed. To construct this command, press F3 to select "Positioning Commands" from the "Step Types" menu shown in Figure X. At this point, the "Positioning Commands" menu as shown in Figure 6 is displayed.

```
               Positioning Function:

                 F1  - Move Incremental
                 F2  - Move Absolute
                 F3  - Gear X to Y
                 F4  - Home Axis
                 F5  - Jog
                 F6  - Stop Motion
                 F7  - Change Speed
                 F8  - Redefine Position
                 F9  - New Max Velocity
                F10  - New Max Acceleration
                ESC  - Cancel Selection
```

Figure 6
EasyACCEL Positioning Function Menu

"Move Incremental" is selected by pressing F2. The ***ACCEL***command "MI" (Move Incremental) is then displayed and the programmer prompted for the distance and acceleration values. These values may be a constant, variable, or expression, and may be entered directly or prompted for as desired by the application programmer. As the values are entered, they are displayed using the proper syntax after the MI. The complete command can therefore be constructed without knowing anything about the programming language or its syntax!

Program Verification

One of the most annoying features of many motion programming languages is that all program de-bugging must be done after the program is downloaded into the motion controller. This makes correcting any programming errors very difficult since the errors must be corrected using the development system and the entire program re-loaded. In addition, unless the motion controller is available during program development, errors are not detected until the motion control system is installed on the machine. This is usually a very time-critical phase of any project and not the place to be finding programming errors!

For this reason, APW provides a built-in Program Verifier to detect programming errors before the application program is downloaded to the MCC. This allows programming errors to be caught and corrected at the programming stage rather than after the control system is installed on the machine.

The Program Verifier feature of APW checks the complete application program for proper syntax, proper use of labels and variables, and proper expression construction. When an error is found, the offending line is displayed with an arrow pointing to the improper character. A message is also displayed indicating what the problem is. Program errors are displayed as they are detected or can be accumulated in a file to provide an error listing. Figure 7 shows a typical syntax error as caught by the Program Verifier.

```
Checking for ACCEL program syntax...
Line 129
MV0+V10@V11
       ^-Expecting comma here!
Press any key...
```

Figure 7
Program Verifier Operation

Automatic Application Program Download

Once the application program is written and verified as described above, it must be downloaded to the MCC for final de-bugging on the machine. APW provides a fully automated utility for downloading the application program and configuring the MCC properly for the application. Downloading the completed program is as simple as selecting an item from the displayed menu.

When the application program has been downloaded to the MCC, APW's Terminal Mode allows the IBM PC to function as a terminal or Operator Control Station for final debugging. Terminal Mode provides a number of pop-up help menus which provide English-language displays of Status Codes, Error Codes, etc. In addition, special diagnostic commands included in ***ACCEL*** allow displaying each command as it is executed and single-stepping the program to make de-bugging easy.

MAKING CUSTOMER SUPPORT EASY

To make supporting the custom application after the machine is delivered to the final customer easy, changes and updates must be able to be made easily. Many times, a bug is discovered only after the machine has been in service for some time. This bug must be fixed and the software and documentation updated and delivered to the customer.

Traditionally, fixing a bug in a custom application program required the programmer to travel to the customer site with a software listing, find the bug, return to the factory to fix it, and send the updated software to the customer. Obviously, this method is not very easy for the customer or the programmer!

Software Updates via Floppy Disk

A better method of updating controls in the field involves providing the customer with an application support software package which runs on an IBM PC. With this approach, software updates can be distributed on inexpensive floppy disks which the customer uses with the application support package to update the controls. Such a support package must be extremely easy-to-use and include only those functions necessary for updating and documenting the application software.

To fulfill the above needs, an application support software package called APPlink has been developed to support ***ACCEL*** applications. APPlink allows the end-user of a custom motion control system to save, restore, and update his custom software as required. Unlike APW, which is used to initially develop the application software, APPlink does not allow the user to directly modify the application software. It simply allows the "compiled" application program, including all setup data and variable definitions and default values, to be uploaded and downloaded between the MCC and the PC . This allows the end user to keep a copy of the custom software for archival purposes and to restore the application to the MCC in the case of a malfunction.

With APPlink, updates to the custom application software may be send directly to the end user on a PC-compatible floppy disk and downloaded to the MCC by the end-user himself. In most cases, a trip by the programmer to the customer site is eliminated completely, reducing both the cost and delay often associated with software updates.

Software Updates via Modem

Alternatively, the applications engineer can communicate directly with the MCC at the customer's site over standard telephone lines using modems. This provides the fastest way to update software in the customer's control.

As shown in Figure 8, the MCC in the customer's factory is connected to the phone system via any standard computer modem. At the other end of the line, the application programmer uses APPlink to download an updated application program directly to the MCC via another modem.

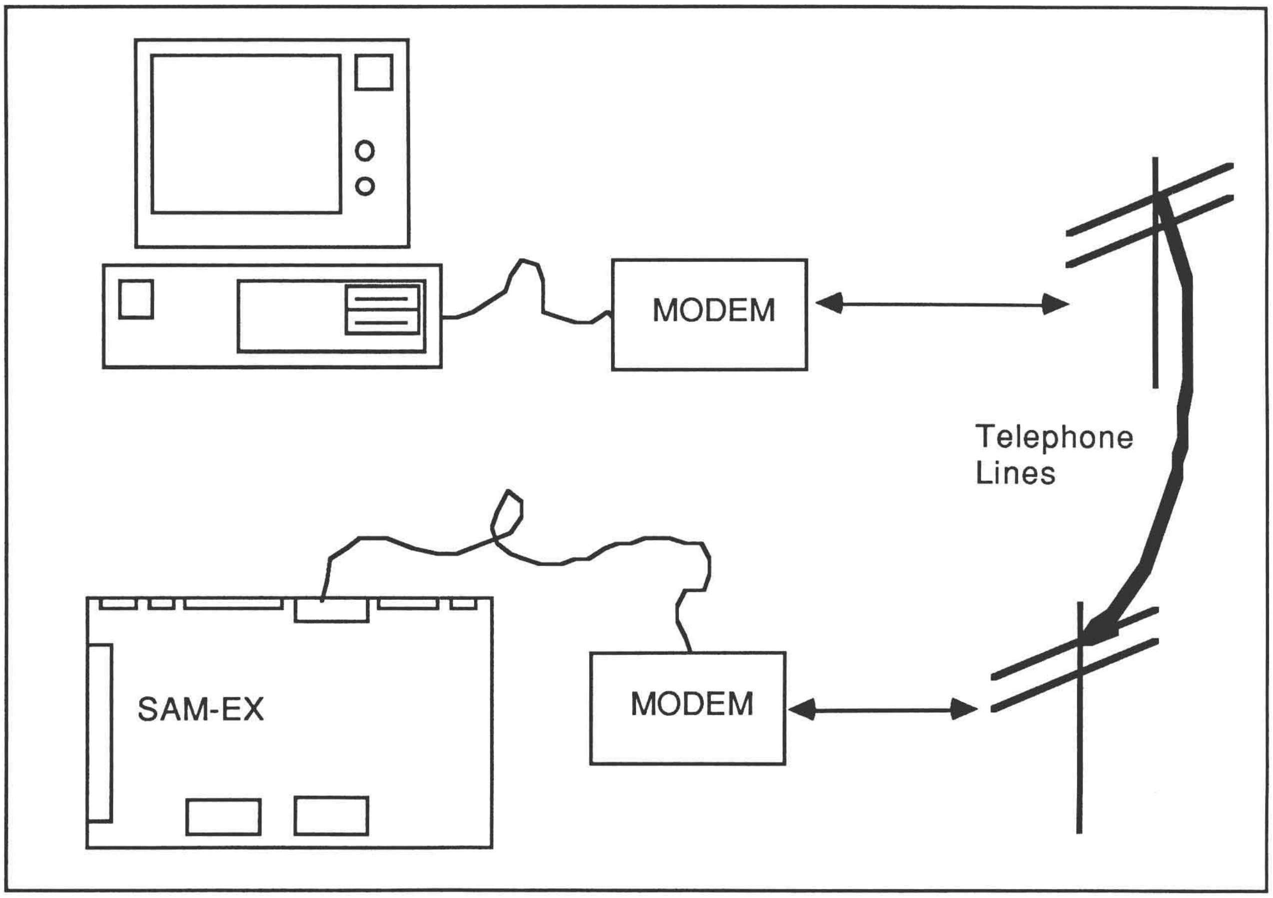


Figure 8
Updating Custom Software Using a Telephone Modem