FORTH in the Computer Numerical Control Environment

John Mullen

Department of Industrial Engineering
Iowa State University
Ames, Iowa

Abstract

In order to help students grasp the concepts of computer-aided manufacture, a small Computer Numerical Control (CNC) workstation was built. It consists of a microcomputer, a vertical mill, a plotter, a printer, and necessary interface electronics. The workstation may be controlled by means of either a FORTH-based control language or a CNC language closely following the standards of EIA RS-358-B.

The development of this workstation has been greatly facilitated by the use of FORTH, which was used to test electronic driver and interface circuits, to control the plotter and mill, to form a basis for the FORTH-like control language, and to implement the EIA standard CNC language. In addition, since the CNC languages are embedded in FORTH, the support features of the FORTH system are available to the user.

The inherent transportability of FORTH combined with a hierarchial modular design results in an implementation that is almost totally hardware-independent and very flexible. This project serves as an extensive demonstration of the utility of the integrated FORTH environment in CNC applications.

Introduction

An important part of any design process is the repeated iterative adaption cycle necessary in which a plan is adjusted to make it work. This important aspect of engineering is not often emphasized in a curriculum because students must master a large number of design techniques and actually implementing designs is both time-consuming and expensive. Since metal-cutting processes are fundamental for industrial engineers, a compact, inexpensive CNC workstation can be used to allow students to gain experience in this important aspect of engineering.

To this end, a compact, self-contained Computer Numerical Control (CNC) workstation was designed and built in the Industrial Engineering Digital Control Laboratory at Iowa State University. This table-top CNC workstation allows students to design parts-cutting programs in an interactive environment and to test the program by cutting parts in machineable wax. In order to make the experience realistic, the control language is modeled after an industry standard and the mill allows three-dimensional contour cutting.

FORTH was instrumental in the design of this sytem and is currently used to operate it. This paper discusses the role of FORTH and some of the techniques employed in the workstation project.

The Workstation

The CNC workstation consists of a Commodore CBM 8032 microcomputer and CBM 8050 disk drive [1] connected to a modified Sherline Vertical Mill [2] by means of a locally designed interface. In addition, a Commodore 2022 printer and a Hewlett-Packard 8228b plotter [3] are interfaced to

the computer to provide program listings and graphical displays of the tool's path. CNC-FORTH, the control language used, is an extension of "FORTH for PET" [4], which was modified with vendor-supplied screens to conform to FORTH-79.

Background

CNC Systems

Computer Numerical Control (CNC) is the use of numbers, letters and symbols to control the action of some machine. Often implied is the concept that each component in the controlled system has a finite number of states determined by the range of the component's travel and by the desired minimum resolution. CNC is used to control cutting equipment, robots, parts insertion equipment, and many other devices [5].

There are several degrees of tool control. In simple point-to-point applications, such as parts-insertion equipment, the tool is moved from one position to the next without undue concern over the intermediate path. However, in attempting to follow a contour, such as in cutting cloth from a pattern, the path must be described exactly. Finally, in the case of some applications, such as metal-cutting, the speed at which the tool moves is also important since tool wear and the part's finish are affected [6].

In this project the controlled device is a small vertical mill with three orthogonal axes. The maximum travel along any axis is seven inches. It was decided that a minimum resolution of about 0.001 in. would be acceptable. In order to provide maximum flexibility, the mill is capable of three-dimensional contouring operations and controllable feed rates in the range of one to ten inches per minute (ipm).

The RS-358-B Standard

While there are a wide variety of CNC languages, most commercial languages conform closely to standard RS-358-B [7]. To make the workstation more realistic, the RS-358-B standard was used as a model for CNC-FORTH.

In this type of language, a program consists of a sequence of blocks. Each block consists of two or more words and is interpreted as a single command. A CNC block is of the form

```
N  G  XYZIJ  F  S  T  M  EOB
```

where

`N`	=	sequence number
`G`	=	preparatory function
`XYZIJ`	=	dimensional data
`F`	=	feed function
`S`	=	speed function
`T`	=	tool function
`M`	=	miscellaneous function
`EOB`	=	end-of-block indicator

The `N` word and the `EOB` terminator are required. All other words are optional but must be in the order shown if present.

Workstation Development

Plan

There were four major tasks. The first was to make the mill computer controllable. The second was to design the interface between the computer and mill. The third was to control the mill's movement and speed, and the last was to implement the CNC language.

Mill Conversion

A Berger-Lahr RDM 63/10 [8] stepper motor is attached to each of the mill's three lead screws. This motor develops 16 in.-oz of torque and has a step size of 9° which produces a minimum

resolution of 0.00125 in. and a maximum feed rate of 17 ipm along the X and Y axes. Gearing is needed on the Z axis to provide sufficient torque and leads to a resolution of about 0.00038 in. and to a maximum feed rate of about 6 ipm along the Z axis. The modified mill is pictured in Fig. 1.

The next question was how to advance the motors. Several options, using FORTH to generate the bit sequences, were tested. On the basis of these tests, half-step operation was shown to be superior in its smoothness and speed. However, positioning was erratic for the odd-numbered phases since only one winding was energized. Thus it was decided to operate the motors in the half-step mode but to halt only at full-step positions.

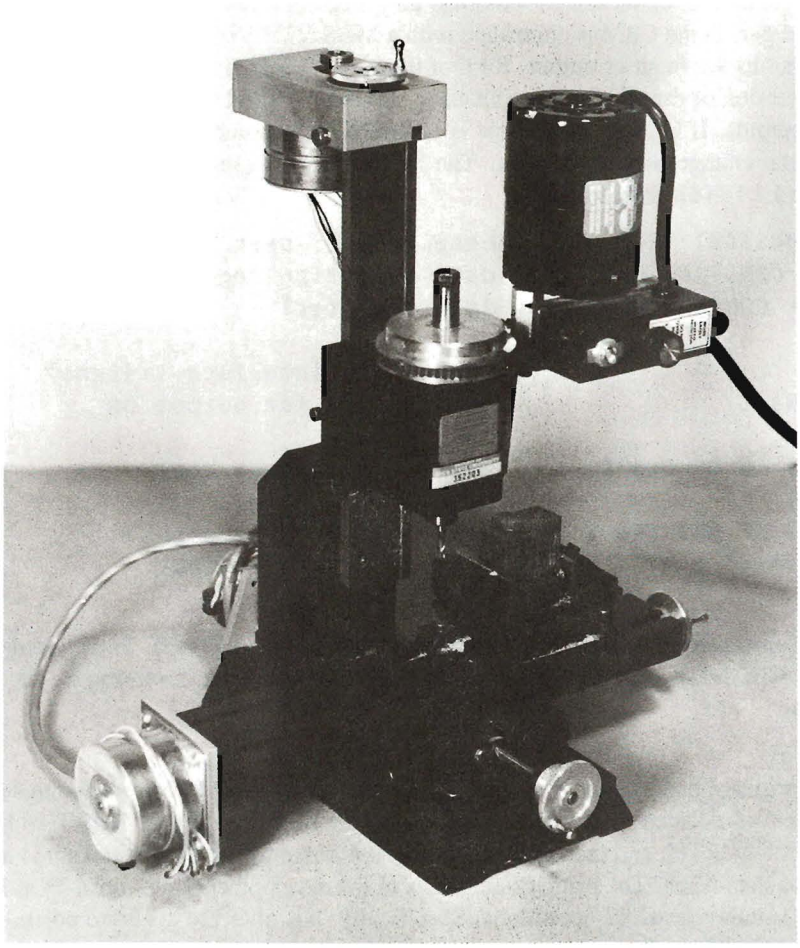

Figure 1. The modified mill.

Interface Design

Each motor requires four bits to control its windings. In order to provide three-dimensional contouring control, all three motors must be operated independently and simultaneously. The primary function of the interface is to permit such control with only six data and one control line of the eight-bit parallel port. A secondary function is to relieve the software of some minor computational tasks. A third function is to permit data input on the same parallel port.

The basic control scheme is to place each motor in one of four possible states: step forward, step back, lock, or free wheel, and then to clock the CB2 control line to cause action. Sequential logic within the interface generates the appropriate series of control signals for each motor.

When CB2 is brought low, the first half step is executed for any motor which is to move. In addition, the command input buffer is latched, which allows the bus signals to be changed in preparation for the next cycle. When CB2 goes high again, the second half step is executed and the command buffer is unlatched.

While CB2 is low and the command buffer is latched, a data-input buffer allows signals from a joystick to be processed as input. This joystick is used to align the tool to the workspace prior to running a program.

Motor control signals are generated by means of three independent circuits. Each circuit steps its motor through the proper sequence of signals to cause forward or reverse half-step motion. Also, any motor's sequence can be locked or windings de-energized (free wheel).

The parallel part in the CBM is controlled with a MOS 6522 VIA, which may be programmed so that any of its bits are input or output. Bit 6 of the parallel port in conjunction with CB2 is used to signal the direction of data flow to the interface. If both bit 6 and CB2 are high, flow is toward the interface (output). If both are low, flow is toward the computer (input). Other combinations permit directional change without conflict. The bus protocol is established through the FORTH words **TALK** and **LISTEN** defined below.

```
59456 CONSTANT VIA        ( base addr of user VIA)
VIA 3 + CONSTANT DDRA     ( data direction register)
VIA 15 + CONSTANT DRA     ( data register)
HEX
: TALK                    ( CBM talks, interface listens)
  7F DDRA C!              ( program VIA for output on
                            bits 0-6)
  DRA C@ 40 OR DRA C! ;   ( set bit 6 of data bus)
: LISTEN
  DRA C@ BF AND DRA C!    ( reset bit 6 of data bus)
  40 DDRA C! ;            ( program VIA for output
                            on bit 6 alone)
```

To prevent unwanted motion CB2 is not changed by **TALK** or **LISTEN**. These words are to be used only when CB2 is low in order to obtain input while the computer is waiting for the motors to complete a step.

Control-FORTH

Once the hardware had been built and tested, the remainder of the project involved designing the software. It was decided that the best approach was first to design a FORTH-like control language, Control-FORTH, and then to design a translator to convert CNC blocks into equivalent Control-FORTH statements. The primary emphasis in the development of Control-FORTH was to control the mill suitably for CNC operations. Specifically, this involved accurate positioning, tool velocity control, point-to-point positioning, and linear interpolation. In addition, both absolute and relative coordinate specification were desired.

Although the development of Control-FORTH is too broad in scope to treat in this paper, the following discussion suggests several key aspects of its implementation.

Stratification. Control-FORTH was written on four levels following Bernstein's concepts of "Software Drivers" [9]. These levels were:

1. Software drivers, which directly manipulate the parallel port.
2. Higher-level drivers, which perform such tasks as coordinate manipulation and linear interpolation calculations.
3. CNC functions, which carry out complete CNC tasks, such as linear interpolation.
4. User-oriented words, which set up the stack as needed for CNC functions to operate.

The joystick is an example of the utility of this appraoch. When the mill was first used, it was very difficult to align the tool to the workpiece. The interface was modified to allow one to send signals to the computer with a joystick. This allows the operator to position the tool precisely from the Mill's position. The changes required to convert the one-way data bus to a two-way data bus were to modify the interface; to define the new words, **TALK** and **LISTEN**; and to redefine two original words, **LOCK** and **UNLOCK**. However, the bulk of the existing software was unaffected.

Coordinate System. In order to function properly, it is imperative that the CNC system always "know" exactly where the tool is. If the tool's position is viewed in terms of motor steps, the X and Y axes are divided into segments of 0.00125 inches each. However, the Z axis is divided into steps of about 0.0003846 inches each. The main problem here is the aspect ratio; that is, a unit distance along the Z axis is not the same as the other two. Thus the calculation of the length of a diagonal and tool speed are complicated. Also, such a coordinate system is implementation-dependent. Thus, if a stepper motor were to be replaced with one of a different step size, all words dependent on the step size would have to be changed. Ideally, a coordinate system should have no aspect ratio problem and should relate to some standard unit of length. However, such a system would not relate to the motors directly, and a cumulative rounding error could arise.

The solution to both aspect ratio and rounding error problems was to develop a dual coordinate system. One part is an orthogonal system with a unit size of 0.0005 in. along each axis. The tool's coordinate is stored in the array **HT-COORD**. The other is a parallel system with a unit size of one step along each axis. The tool's coordinate for the latter system is stored in **S-COORD**. The two are related by the word **HT->S**.

```
ARRAY STEPS/TURN 40 , 40 , 130 ,
: HT->S ( n bas - n) STEPS/TURN @ 100 */;
```

When a move is specified, the destination is first calculated in the half-thousandths system. Then the nearest equivalent point in the stepper system is found, and the relative distance in steps from the current motor's position to the new one is calculated. Once the conversion is made, the move is executed.

This scheme eliminates cumulative rounding error if coordinates are specified as multiples of 0.0005 in., regardless of the step size. Also, if a motor is replaced, one need only change the appropriate cell of **STEPS/TURN**.

The major Control-FORTH words are listed in Table 1. These words may be used to control the mill or plotter. There is also the set of words defined in Table 2 that is used only with the plotter.

An Example. The following program segment will cut a 1.0 × 0.5 in. slot with its bottom at 0.20 in. below the x-y plane. The tool diameter is ⅜ in. The initial tool position is at 0.5 in. above the x-y plane and some unspecified x-y coordinate. The slot's lower left-hand corner is at (0,0), and the lower right at (1,0). The cut is to be made at 10 ipm.

```
375 375 1000 ABS PTP     ( move to [0.1875,0.1875,0.5])
50 FEEDRATE !            ( set feed rate to 5 ipm)
375 375 -400 LIN         ( cut to [0.1875,0.1875,-0.2])
100 FEEDRATE !           ( set feed rate to 10 ipm)
1625 375 -400 LIN        ( cut to [0.8125,0.1875,-0.2])
1625 625 -400 LIN        ( cut to [0.8125,0.3125,-0.2])
 375 625 -400 LIN        ( cut to [0.1875,0.3125,-0.2])
 375 375 -400 LIN        ( return to starting corner)
0 0 1400 REL PTP         ( lift tool clear of work)
```

CNC-FORTH

The final phase of this project was to develop words that would translate standard CNC blocks into Control-FORTH commands. There were four distinct steps in this phase. The first was to develop an overall plan. The second task was to develop necessary components to interpret CNC words. The third step was to systematically develop the meanings of each word in CNC-FORTH, and the final step was to define words that would actually translate the interpreted information into action. Representatives of the screens which define CNC-FORTH are in the Appendix, and key details of its implementation are discussed below.

Basic Plan. The fundamental idea was suggested by Bernstein [10]. In his approach, the first word of each command causes a special vocabulary to be made current; then, each intermediate word changes variables as needed, and the final word causes action. Bernstein's approach was modified so that one enters the CNC vocabulary at the start of the CNC program (screen 156) but action occurs at the end of each block. Each word between the **N**-word (screen 162) and **EOB** (screen 194) is a FORTH word that affects some variable. The terminator, **EOB**, carries out the action indicated by the variable settings at the end of the block.

There are three major advantages to this approach. First of all, since each word is a FORTH word, the FORTH interpreter can be used to interpret the block. Thus, CNC-FORTH is an extension of FORTH-79 and Control-FORTH, not a separate program. Secondly, since all words except **EOB** affect only variables, one can verify their effect by examining the variables without risking possibly dangerous action, such as the tool cutting into the mill's bed or fixtures. Screens 152 and 153 define the variables used to represent the components of a block. The word **[EOB]** (Screen 164) is a dummy version of **EOB** that displays these variables. **CNC?** (Screens 157 and 158) can be used to display the variable set at any time. A third advantage is that, since these words do not affect any external devices, they can be used to set up motion for either controlled device.

CNC Components. There are three major problems at this level. First of all, the CNC language is a post-fix language; that is, the operands follow their identifiers. The second problem is that operands may be decimal fractions, such as "3.75" or "−0.0015," which may represent inches, inches per minute, millimeters, or some other context-dependent value. The final problem is the enforcement of word order within a block.

The word **GET-NUMBER**, defined below, will place a number that follows it in the input stream on the stack. The position of the decimal point (if any) is then placed above it. This word is used to obtain numbers by every word requiring a numerical operand.

```
: GET-NUMBER ( - d n) 32 WORD  NUMBER DPL @ ;
```

Other words then convert the double number into an appropriately scaled single number. For example, **#CONVERT** (screen 169) will change an input value (d) to half-thousandths of an inch. **SCALE-FACTOR@** compares the position of the decimal point (n2) to the limit on range (n3); then it retrieves the multiplier indicated by n2. The value of n1 will determine whether conversion is from inches or millimeters.

```
: #CONVERT ( d n1 n2 - n) SCALE-FACTOR@ >R D->S  R> CFACT1 @ */ ;
```

In order to enforce standard word order, a command level is assigned to each class of words. The level for an **N**-word is zero; that of an **M**-word is one; and so on. **N** sets the value of **CMD-LVL** to zero. Thereafter, each word uses **?LVL!** to verify that its level is greater than that of the last word used. If it is, **?LVL!** stores the level of the current word in **CMD-LVL**. If not, it displays an error message and exits through **CNC-ERR**, which identifies the block containing the error and stops the program.

```
: ?LVL! ( n) CMD-LVL @ OVER < IF CMD-LVL !
 ELSE ." WORDS OUT OF ORDER" CNC-ERR THEN ;
```

Finally, if an axis is not mentioned in a command, its coordinate value is not to be changed. A flag array indicates whether or not each axis is referred to in a current block. If an axis is not mentioned, its current value is taken as the desired value (screens 185 and 186).

CNC Word Definitions. A distinction must be made at this point. Some words, such as **G01** and **M05**, are codes; that is, the letter and number together have a unique meaning. Other words, such as **N** and **X**, require operands. Codes must be entered as shown, i.e., without spaces, whereas there must be a space between such words as **N** and their operands.

Codes are words that simply set a flag or variable that indicates a context or action at the end of the block. For example, **G01** (screen 174) sets **DEF-MVCODE** (screen 178) to 3, which indicates linear interpolation. **M05** sets a flag that will cause execution to pause in the block and then display the message "turn spindle on."

Words with operands also set variables and flags but in addition must use **GET-NUMBER** and possibly **#CONVERT** to obtain and process the operand. Operands are checked for appropriate range and then stored in the proper variable.

CNC Action. The word **EOB**, defined in Screen 194, marks the end of each CNC block. It carries out the action prescribed by the CNC variables. Since this word causes motion, a different version is needed for each device. Note in its definition that vocabularies are switched after its header is established so that this definition may be used to define **EOB** in both the **MILL** and **PLOTTER** vocabularies.

Aside from carrying out the action required by a block, **EOB** also checks to see if a stop has been indicated by a word, such as **M05**, within the block, if the step or trace mode has been set, and finally if the stop button has been pressed.

The words shown in Table 3 implement a minimal subset of RS-358-B, which is sufficient to allow writing fairly complex parts programs. Table 4 lists words used to control the system. Finally, Table 5 lists words that extend the RS-358-B standard.

An Example. The following program performs the same task as the earlier example of Control-FORTH:

```
N 10 G70    EOB                  ( input in inches)
N 20 G90    EOB                  ( absolute coordinates)
      .
      .
      .
N 500 G00   X .1875   Y .1875   EOB   ( position over first corner)
N 510 G01   Z -.2     F  5      EOB   ( enter work)
N 520       X .8125   F 10      EOB   ( cut to 2nd corner)
N 530       Y .3125             EOB   ( 3rd corner)
N 540       X .1875             EOB   ( 4th corner)
N 550       Y .1875             EOB   ( back to first)
N 560 G00   Z .5                EOB   ( lift tool clear)
```

Conclusion

This author has programmed extensively in several assembly languages as well as in numerous versions of BASIC and FORTRAN. Although he had never used FORTH before, he was able to learn how to use the language and complete this project more easily than if he had used the other languages. The combination of features available in FORTH makes it a much more suitable language than any of the others for this type of application. Thus FORTH is very well suited to the CNC environment.

At this time, the Iowa State University Industrial Engineering Department is considering the adaptation of this project to another computer, possibly a Texas Instruments Professional. If this project had been done in BASIC, its adaptation to the TI would be quite difficult. However, since FORTH-79 is available for the TI, the adaptation should be straightforward. While portions of the current application are in assembly language, the TI's microprocessor is quite fast, so the FORTH versions of these words could be used.

The utility of FORTH in this environment, together with its transportability, suggests great promise in FORTH as a means of implementing CNC languages. There is also a possibility that it can provide a basis for an exchange of ideas and algorithms in this field if a standard of media exchange can be established.

The project is continuing. At this time, the system is being tested by students to determine how its utility might be improved. Those wishing further information are referred to the author's thesis [11] or may contact the author at the address above.

Acknowledgments

The Department of Industrial Engineering at Iowa State University is sponsoring this project. ISU's Engineering Research Institute has also provided technical assistance. Dr. Roger W. Berger suggested this project and contributed his support towards its completion; his assistance is also gratefully acknowledged.

References

1. Commodore Business Machines, Norristown, PA.
2. Sherline Products, San Marcos, CA.
3. Hewlett-Packard Co., 16399 W. Bernardo Drive, San Diego, CA.
4. A B Computers, 252 Bethlehem Pike, Colmar, PA.
5. Pressman, R. S. and Williams, J. E., *Numerical Control and Computer Aided Manufacturing,* John Wiley and Sons, New York, 1977, Chap. 1.
6. Childs, J. J., *Principles of Numerical Control,* 3rd ed., Industrial Press, Inc., New York, 1982, Chap. 2.
7. Childs, J. J., *Principles of Numerical Control,* 3rd ed., Industrial Press, Inc., New York, 1982, pp. 53-77.
8. BERGER-Lahr Corp., Fitzgerald Drive, Jaffrey, NH.
9. Bernstein, M., "FORTH in the computer toolbox," *FORTH Dimensions,* Vol. 4, No. 2, July/August 1982, pp. 6-8.
10. Bernstein, M., "Stepper Motor Control: A FORTH Approach," *MICRO,* No. 45, February 1982, pp. 95-99.
11. Mullen, J., "How FORTH May Be Used to Implement an EIA Standard CNC Language, Demonstrated by Means of a Small, Microcomputer-controlled Vertical Mill," Master's thesis, Iowa State University, Ames, Iowa, 1984.

Manuscript received May 1985

John Mullen received a B.A. degree in Mathematics from the University of Pennsylvania in 1968 and a M.S. degree in Industrial Engineering from Iowa State University in 1984. He is currently an instructor pursuing a Ph.D. in Industrial Engineering at ISU. His interests include operations research, simulation, and digital computer control.

Appendix

Table 1. Control-FORTH general words.

ABS	select absolute coordinate reference
ABS-LIN	(x y z) move to (x,y,z) along a straight line
ABS-PTP	(x y z) move to (x,y,z) as quickly as possible, not necessarily along a straight line
ALL-ZERO	sets coordinates to (0,0,0)
CUT	select the mill
DRILL	(n1 n2) drills hole of to n2 or of depth n2, depending on reference mode. Dwells n1 msec at bottom.
JOY-MOVE	control the mill with the joystick
LIN	(n1 n2 n3) functions as ABS-LIN or REL-LIN depending upon reference mode
LOCK	lock tool in current position
PLOT	select plotter
POLY	(xk yk zk ... x1 y1 z1 k) move through the k points indicated
POS?	display current coordinate position
PTP	(x y z) function as ABS-PTP or REL-PTP depending on reference mode
REL	select relative or incremental reference
REL-LIN	(x y z) move from current position (x0,y0,z0) to (x0+x,y0+y,z0+z) along a straight line
REL-PTP	(x y z) move from current position (x0,y0,z0) to (x0+x,y0+y,z0+z) as rapidly as possible
SET-COORD	(x y z) define current coordinate to be (x,y,z)
SET-RATE	(n) set feed rate to n tenths of an inch per minute
SLOT	(n x3 y3 z3 x2 y2 z2 x1 y1 z1) cut a surface using SURFACE, then make a final pass around the perimeter
SURFACE	(n x3 y3 z3 x2 y2 z2 x1 y1 z1) cuts the surface defined by the current point and the three other points in n+1 passes
UNLOCK	de-energizes the mill's actuators, even if the plotter is selected
X	(− 0) index value for x-axis
XYPOLY	(xk yk ... x1 y1 k) a two-dimensional version of POLY
XYSLOT	(n x3 y3 x2 y2 x1 y1) a two-dimensional version of SLOT
XYSURFACE	(n x3 y3 x2 y2 x1 y1) a two-dimensional version of SURFACE
Y	(− 1) index value for Y-axis
Z	(− 2) index value for Z-axis
ZERO	(n) will zero the axis whose index value is n

```
Screen # 152
    00  ( fundamental cnc variables                                  4-26-84)
    01  cnc-voc definitions
    02
    03  variable   cnc-block   ( id number of the current cnc block    )
    04
    05  variable   cmd-lvl     ( used to check order of cnc words      )
    06
    07  array      cfact1       ( used to scale operands               )
    08   10000 ,   1000  ,   10  ,    10  ,    1  ,
    09
    10  variable   cfact2       ( used to convert operands into half-  )
    11                          ( thousandths.  if input is in inches,  )
    12                          ( cfact2 = 5, if mm, it is 127          )
    13
    14  variable   need-dwell   ( flag:  true -> x will get dwell       )
    15               -->

Screen # 153
    00  ( additional variables for cnc words                          4-26-84)
    01  cnc-voc definitions
    02
    03  variable  mvcode              ( indicates the type of motion   )
    04
    05  variable  motion              ( flag: true -> motion in this   )
    06                                (       cnc block                )
    07
    08  variable  def-mvcode          ( default type of motion         )
    09
    10  variable  stop                ( indicates stop or end          )
    11
    12  array     arg       20 allot  ( room for up to ten arguments   )
    13
    14  array     arg-used  20 allot  ( true -> arg used in current blk)
    15

Screen # 156
    00  ( cnc  sets up the system for cnc operation                   4-26-84)
    01  forth definitions
    02  : cnc
    03        definitions
    04        cnc-voc
    05        [compile] cnc-voc
    06        -1 feedrate !     ( these variables are set to bad values)
    07         0 cfact2   !     ( so that errors due to the ommission  )
    08        -1 mvcode   !     ( of codes may be detected              )
    09        -1 abs-ref  !
    10        -1 def-mvcode !
    11         0 cnc-block !    ( initializations for the first block  )
    12        -1 cmd-lvl   ! ;
    13
    14
    15
```

```
Screen # 157
    00  ( cnc?   final version                                    4-26-84)
    01  forth definitions decimal
    02  : cnc?
    03    cnc-voc
    04    ." context: " .context 5 spaces
    05    ." current: " .current
    06    ."      cnc-block: " cnc-block @ 5 .r  cr
    07    ."     cmd-lvl: " cmd-lvl @ 4 .r
    08    ."      cfact2: " cfact2 @ 4 .r
    09    ."      mvcode: " mvcode @ 4 .r
    10    ."  def-mvcode: " def-mvcode @ 4 .r  cr
    11    ."     abs-ref: " abs-ref @ 4 .r
    12    ."        stop: "   stop   @ 4 .r
    13    ."    feedrate: " feedrate @ 4 .r
    14    ."      motion: " motion @ .f cr
    15                    -->

Screen # 158
    00  ( cnc?                  continued                         4-26-84)
    01
    02     10 spaces
    03     10 0 do i  6 .r loop cr
    04     ." arg:      "
    05     10 0 do i arg @  6 .r loop cr
    06     ." arg-used:  "
    07     10 0 do i arg-used @  .f ."  " loop ;
    08
    09
    10
    11
    12
    13
    14
    15

Screen # 162
    00  ( n     first word in each block                          5-25-84)
    01  cnc-voc definitions decimal
    02
    03  : n
    04      0 ?lvl!                    ( n must be the first word        )
    05      get-number                 ( get following number            )
    06      -1 >  if                   ( if dpl > -1,                     )
    07        .' block number may not contain a decimal point' cr
    08        n-err
    09      endif
    10      drop                       ( convert to a single number      )
    11      dup 1 < over 999 > or if ( if the number out of range      )
    12        . .' cannot be used as a block number'
    13        n-err
    14      else
    15                      -->
```

```
Screen # 163
 00  ( n        continued                                        5-25-84)
 01
 02         cnc-block !              ( store block number              )
 03         false motion      !      ( initialize certain variables    )
 04         0      mvcode     !
 05         0      stop       !
 06         false need-dwell !
 07         10 0 do                  ( reset arg flags                 )
 08           false i arg-used !
 09         loop
 10       endif ;
 11
 12
 13
 14
 15

Screen # 164
 00  ( [eob]  test version of eob                              4-26-84)
 01
 02
 03  : [eob]
 04    cnc-voc
 05    cmd-lvl @ 0< if        (   if the cmd-lvl is negative,        )
 06      ." missing n word"   (     start of error message          )
 07      cnc-err              (     rest of message and abort       )
 08    endif
 09    cr cnc?                (   display the cnc variables         )
 10    -1 cmd-lvl !           (   set cmd-lvl negative for n word   )
 11    10 10 10 rel-ptp       (   jog device a little               )
 12    -10 -10 -10 rel-ptp
 13    cr pos? ;              ( display current coordinates         )
 14
 15

Screen # 169
 00  ( scale-factor@   dpl limit --- scale-factor             5-27-84)
 01  cnc-voc definitions
 02
 03  : scale-factor@
 04    over over > if         ( if dpl exceeds the limit            )
 05      .' more than ' . .'  decimal places' cnc-err
 06    endif
 07    drop                    ( toss the limit                      )
 08    0 max                   ( -1 and 0 mean the same here         )
 09    cfact1 @ ;
 10
 11
 12
 13
 14
 15
```

```
Screen # 172
    00  ( arg!  n i ---                                          4-26-84)
    01  cnc-voc definitions
    02
    03  : arg!
    04      dup arg-used @ if       ( if the argument has already been )
    05                              ( used in this block,              )
    06        .' argument ' .       ( identify the argument number     )
    07        .'  has been used twice'
    08        cnc-err
    09      endif
    10      swap over               ( n i --- i n i                    )
    11      arg !                   ( store the value                  )
    12      true swap               (   i --- true i                   )
    13      arg-used ! ;            ( mark argument as used            )
    14
    15

Screen # 173
    00  ( -xyz            x  y  z                                  5-27-84)
    01  cnc-voc definitions decimal
    02  : -xyz                  ( common element of dimension words   )
    03      ?lvl!               (   set / check command level         )
    04      cfact2@             (   get conversion factor             )
    05      get-number
    06      4 #convert          (   convert number to half-thousandths)
    07      true motion ! ;     (   set the motion flag               )
    08  : y     3 -xyz      1 arg! ;
    09  : z     4 -xyz      2 arg! ;
    10  : x     need-dwell @ if  (   if dwell is needed               )
    11      2 ?lvl!  get-dwell
    12    else
    13      2 -xyz
    14    endif
    15    0 arg! ;

Screen # 174
    00  ( g0, g01, g70, g71, g90, g91, g93                      4-04-84)
    01  cnc-voc definitions decimal
    02  : g00                              ( point to point motion     )
    03      1 ?lvl!                        (   command level is one    )
    04      2 def-mvcode ! ;               (   motion code is two      )
    05  : g01  1 ?lvl!  3 def-mvcode ! ;   ( linear interpolation      )
    06
    07  : g70  1 ?lvl!    5 cfact2   ! ;   ( inch input                )
    08  : g71  1 ?lvl!  127 cfact2   ! ;   ( millimeter input          )
    09
    10  : g90  1 ?lvl!    1 abs-ref  ! ;   ( absolute input            )
    11  : g91  1 ?lvl!    0 abs-ref  ! ;   ( incremental input         )
    12
    13  : g93  1 ?lvl!    1 mvcode   ! ;   ( preload registers         )
    14
    15
```

```
Screen # 178
    00  ( m00, m02, m03, m05, m06                                    4-04-84)
    01  cnc-voc definitions decimal
    02
    03  : -m                          ( common factor                    )
    04    13 ?lvl!                    (   set level to ten               )
    05    stop !                      (   store code in stop             )
    06    cr ;                        (   carriage return                )
    07
    08  : m00
    09      2 -m
    10      ." program stop" ;
    11
    12  : m02  3 -m  ." program end"     ;
    13  : m03  1 -m  ." turn spindle on. "  ;
    14  : m05  1 -m  ." turn spindle off. " ;
    15  : m06  1 -m  ." tool change"        ;

Screen # 185
    00  ( xyz-arg@                                                   4-04-84)
    01  cnc-voc definitions decimal
    02  : xyz-arg@
    03      abs-ref @
    04      begin-cases
    05        1 case                    ( case of absolute motion        )
    06          3 0 do
    07            i arg-used @ if       (   if the argument was used,    )
    08              i arg @             (     get it                      )
    09            else                  (   if it was not,                )
    10              i ht-coord @        (     get the current coordinate)
    11            endif
    12          loop
    13        end-case
    14        0 case                    ( case of incremental motion     )
    15              -->

Screen # 186
    00  ( xyz-arg@ continued                                          4-04-84)
    01
    02          3 0 do
    03            i arg-used @ if
    04              i arg @
    05            else
    06              0
    07            endif
    08          loop
    09          end-case
    10        else-case
    11          .' reference method not specified'
    12          cnc-err
    13          end-case
    14      end-cases ;
    15
```

```
Screen # 187
    00  ( cnc-stop   stops execution of a cnc program          5-25-84)
    01  cnc-voc definitions
    02
    03  : cnc-stop
    04      current @        ( restore the original context vocabulary )
    05      context !
    06      cnc              ( set up for new program                   )
    07      quit             ( stop interpreting the program screen     )
    08      ;
    09
    10
    11
    12
    13
    14
    15

Screen # 194
    00  ( eob                                                  5-25-84)
    01
    02  : eob
    03    cnc-voc
    04    chk-cmd-lvl   ( verify syntax and signal block end           )
    05    cnc-motion?   ( set up stack if motion is to occur           )
    06    cnc-move?     ( carry out motion, if there is any            )
    07    cnc-stop?     ( check for a programmed stop or pause         )
    08    break? ;      ( check for an external stop or pause, such as )
    09                  ( an operator's interrupt or single-step mode  )
    10
    11
    12
    13
    14
    15

Screen # 210
    00  ( cnc-move?      expanded version                      5-30-84)
    01  : cnc-move?
    02      cnc-voc
    03      -dup if          ( if there is to be motion,               )
    04      begin-cases
    05        1 case             set-coord                  end-case
    06        2 case             ptp                        end-case
    07        3 case             lin                        end-case
    08        4 case  drop drop  dwell                      end-case
    09        5 case  -drill     drill                      end-case
    10        6 case  -poly      poly                       end-case
    11        7 case  -cpoly     poly      abs-lin          end-case
    12        8 case  -surface   surface                    end-case
    13        9 case  -surface   slot                       end-case
    14        else-case .' undefined motion'    cnc-err     end-case
    15      end-cases  endif ;
```