

Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292333
Fax: +44 (0)1684 297929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412.968.9744
Fax: +1 412.968.9746

Tomson Centre
118 Zhang Yang Rd., B1701
Pudong New Area, Shanghai,
Postal code: 200122
P. R. CHINA
Tel/Fax: +86-21-58797659



Doc No.: AN-208

Version: 1.0

Date: 20 May 2008

Subject: Using the GCODE program functions

APPLICATION NOTE

www.triomotion.com

1. Target user profile

The intended use for the GCODE operation is to enable specialist machine builders to use Trio's Motion Coordinator. It provides solutions for the control of X,Y or X,Y,Z axes where the motion program source is from a CAD/CAM program that outputs GCODE sequences.

This GCODE implementation is not intended to replace full-functioning CNC controllers on sophisticated metal turning/milling machines.

Example applications include; woodcarving, laser cutting, glue laying, milling, routing, PCB drilling.

2. The Motion Coordinator GCODE implementation

GCODE processing has been implemented on some special versions of Motion Coordinator system software. The system allows up to 3 axes to be controlled by a sequence of GCODE commands that are resident in one or more of the programs saved to the Motion Coordinator.

Trio Motion Technology's preliminary G Code implementation has been separated into three parts:

1. G Code file handling
2. G Code tokeniser and parser and
3. TrioBASIC programme that dispatches the G Code instructions

This implementation was chosen to aid flexibility of CNC operation. A typical application would require a startup program, written in BASIC and the CGODE dispatcher program which is also a BASIC language program. This leaves space for up to 12 GCODE programs that can be loaded into the memory and can be selected for running as required.

3. System requirements

Motion Coordinator with GCODE system software. (contact Trio for details)

Windows PC with Motion Perfect and/or the Trio Autoloader software.

CAD/CAM software to output a GCODE sequence as a file.

Motors, drives and mechanical assembly to provide the required motion.

4. Overview

GCODE files have the windows file extension .gco. gco files can be loaded into the Motion Coordinator's user program area, where they will be identified by the operating system as GCODE programs. As a GCODE program, they can be run, stepped and stopped from the Motion Perfect control panel and can be run or stopped by TrioBASIC programs using the RUN and STOP commands.

Each line of a GCODE program is parsed by the system software and the parameters from the line are passed to the GCODE dispatcher program as values in global (VR) variables. It is the GCODE dispatcher program, written in BASIC, which controls the final action of each line of the GCODE.

The following GCODE tokens are supported;

G, M, F, S, T, X, Y, Z.

The G Code has access to all the TrioBASIC mathematical functions, including global VR variables. It has no local variables.

The G Code is fully interpreted so during the compilation phase, no compiled code is generated. The parser fully validates the G Code program so that the dispatcher does not have to handle syntax errors. Unknown tokens are passed to an unknown token handler routine in the Dispatcher program. Also there is no special handling of the end of block states in the G Code parse, but this information is relayed to the Dispatcher programme.

5. GCODE Dispatcher

The dispatcher is a TrioBASIC program that controls the action of each GCODE token type. This program is provided by Trio Motion Technology, but can be edited by the user to provide specialised functions as required.

5.1. Token passing

The command token and all its parameters are passed to the Dispatcher program via the global VR variables. The start of this variable block is set by the system parameter GCODE_BASE. The VR addressed by GCODE_BASE contains the GCODE command and the following array of VRs contains the parameters associated with that command.

```
GCODE_BASE=500
```

```
' initialise program variables
GLOBAL "G_COMMAND",GCODE_BASE
GLOBAL "A_VALUE",GCODE_BASE+1
GLOBAL "B_VALUE",GCODE_BASE+2
GLOBAL "C_VALUE",GCODE_BASE+3
GLOBAL "D_VALUE",GCODE_BASE+4
GLOBAL "F_VALUE",GCODE_BASE+6
GLOBAL "G_VALUE",GCODE_BASE+7
GLOBAL "I_VALUE",GCODE_BASE+9
GLOBAL "J_VALUE",GCODE_BASE+10
GLOBAL "K_VALUE",GCODE_BASE+11
GLOBAL "M_VALUE",GCODE_BASE+13
GLOBAL "P_VALUE",GCODE_BASE+16
GLOBAL "R_VALUE",GCODE_BASE+18
GLOBAL "S_VALUE",GCODE_BASE+19
GLOBAL "T_VALUE",GCODE_BASE+20
GLOBAL "X_VALUE",GCODE_BASE+24
GLOBAL "Y_VALUE",GCODE_BASE+25
GLOBAL "Z_VALUE",GCODE_BASE+26
```

5.2. Dispatch G codes

If the G_COMMAND is “G”, then the dispatch program calls a subroutine called dispatch_g. This contains all the motion sequences for running G command tokens; for example to run the command G0, (rapid positioning) the following sequence is provided:

```
IF g_value = 0 THEN ' rapid positioning
  GOSUB normalize
  setspeed=SPEED
  SPEED=max_velocity
  WAIT IDLE 'wait for any interpolated moves to finish
  MOVE(x,y,z)
  WAIT IDLE
  SPEED=setspeed
```

5.3. Dispatch M codes

When the G_COMMAND value is ASCII “M”, the dispatcher calls the dispatch_m subroutine. The actions of various M codes are likely to be very different from one machine to another and it is this routine that it is most convenient to be changed by the user. For example the M code for “Flood Coolant On” is M8. This could simply be an action to turn on a digital output which is connected to a solenoid valve:

```
ELSEIF m_value=8 THEN
  PRINT #5,"Flood coolant ON"
  OP(10,ON)
```

5.4. Dispatch F codes

The only F code implemented is the Feed Rate:

```
dispatch_f: ' feed rate
  SPEED=f_value
  RETURN
```

5.5. Dispatch S codes

The S code is for setting the spindle speed. A suggested way to do this might be to set the voltage on an analogue output of the Motion Coordinator. For example:

```
dispatch_s: ' spindle speed
  PRINT #5,"Setting spindle speed: ";s_value[0]
  DAC AXIS(3)=s_value * 3.2768
  RETURN
```

5.6. Dispatch T codes

T codes are for Tool Change. There is no suggested program sequence to do this as it will vary to a great extent between applications. Whenever a T command token is parsed by the system software it is passed to a routine in the Dispatcher. This routine must be populated with the appropriate sequences for tool changing. The framework for T command processing would look like this:

```
dispatch_t: ' tool selection
  PRINT "Select Tool #";t_value[0]
  IF t_value=1 THEN
    ' tool #1 sequence
  ELSEIF t_value=2 THEN
    ' tool #2 sequence
  ELSEIF t_value=3 THEN
    ' tool #3 sequence
  ELSE
    ' "no tool available" sequence
```

RETURN

5.7. The normalize routine

A GCODE program might require the Motion Coordinator to operate with either absolute or with incremental position values. Each move that is sent to the Dispatcher is filtered through the “normalize” subroutine so that absolute positions can be converted to relative distances when required. This routine should not be changed by the user.

6. Limitations

The present GCODE implementation can perform interpolated 2 or 3 axis moves. The move types available are limited to those which exist in the TrioBASIC language. This includes circular interpolation in 2 dimensions.

3D surface interpolation is not available.

Tool offset functions are not included in this implementation, but these can be provided by the CAD/CAM package which generates the GCODE program.

Specifications and limitations may change without notice. For the latest information, please contact Trio Motion Technology when assessing the suitability of this GCODE implementation for your application. Our application engineers will be happy to assist and advise you about ways to tailor the GCODE dispatcher to suit you.

7. GCODE Dispatcher program

Here is the full listing of an example GCODE Dispatcher program.

```
' initialise the parser
GCODE_BASE=500
g_command=0

' initialise program
GLOBAL "G_COMMAND",GCODE_BASE
GLOBAL "A_VALUE",GCODE_BASE+1
GLOBAL "B_VALUE",GCODE_BASE+2
GLOBAL "C_VALUE",GCODE_BASE+3
GLOBAL "D_VALUE",GCODE_BASE+4
GLOBAL "F_VALUE",GCODE_BASE+6
GLOBAL "G_VALUE",GCODE_BASE+7
GLOBAL "I_VALUE",GCODE_BASE+9
GLOBAL "J_VALUE",GCODE_BASE+10
GLOBAL "K_VALUE",GCODE_BASE+11
GLOBAL "M_VALUE",GCODE_BASE+13
GLOBAL "P_VALUE",GCODE_BASE+16
GLOBAL "R_VALUE",GCODE_BASE+18
GLOBAL "S_VALUE",GCODE_BASE+19
GLOBAL "T_VALUE",GCODE_BASE+20
GLOBAL "X_VALUE",GCODE_BASE+24
GLOBAL "Y_VALUE",GCODE_BASE+25
GLOBAL "Z_VALUE",GCODE_BASE+26
absolute=TRUE

' initialise system
imperial_units=1000
metric_units=1000
x_value=0
y_value=0
```

```

z_value=0
x_pos=0
y_pos=0
z_pos=0
max_velocity=300

BASE(0)
MERGE=OFF

WHILE TRUE
  ' wait for a command
  WAIT UNTIL g_command

  ' dispatch command
  GOSUB dispatch_block

  ' clear command
  g_command = 0
WEND

dispatch_block:
  ' --- DISPATCH
  ' end of programme?
  IF -2 = g_command THEN
    WAIT IDLE
  ' endif of line
  ELSEIF -1 = g_command THEN
    ' F
  ELSEIF 70 = g_command THEN
    GOSUB dispatch_f
  ' G
  ELSEIF 71 = g_command THEN
    GOSUB dispatch_g
  ' M
  ELSEIF 77 = g_command THEN
    GOSUB dispatch_m
  ' S
  ELSEIF 83 = g_command THEN
    GOSUB dispatch_s
  ' T
  ELSEIF 84 = g_command THEN
    GOSUB dispatch_t
  ELSE
    PRINT "unexpected token ";CHR(g_command);" ";g_command[0]
  ENDIF
RETURN

dispatch_f: ' feed rate
  SPEED=f_value
  RETURN

dispatch_g: ' go
  IF g_value = 0 THEN ' rapid positioning
    GOSUB normalize
    PRINT "MO(";x;" ";"y;" ";"z;")"
    setspeed=SPEED
    SPEED=max_velocity
    WAIT IDLE 'wait for any interpolated moves to finish
    MOVE(x,y,z)

```

```

WAIT IDLE
SPEED=setspeed
ELSEIF g_value=1 THEN ' linear move
GOSUB normalize
PRINT "MO(";x;"",";y;"",";z;"")"
MOVE(x,y,z)
ELSEIF g_value=2 THEN ' clockwise circular move
GOSUB normalize
PRINT "MC(";x;"",";y;"",";x+i_value;"",";y+j_value;"",0)"
MOVECIRC(x,y,x+i_value,y+j_value,0)
ELSEIF g_value=3 THEN ' clockwise circular move
GOSUB normalize
PRINT "MC(";x;"",";y;"",";x+i_value;"",";y+j_value;"",1)"
MOVECIRC(x,y,x+i_value,y+j_value,1)
ELSEIF g_value=4 THEN ' cutter compensation off
WAIT IDLE
WA(p_value)
ELSEIF g_value=40 THEN ' cutter compensation off
radius = 0
ELSEIF g_value=41 THEN ' cutter compensation left
radius = -d_value
IF radius THEN PRINT "WARNING: Cutter compensation not implemented"
ELSEIF g_value=41 THEN ' cutter compensation left
radius = d_value
IF radius THEN PRINT "WARNING: Cutter compensation not implemented"
ELSEIF g_value=70 THEN ' imperial mode
UNITS AXIS(0)=imperial_units
UNITS AXIS(1)=imperial_units
UNITS AXIS(2)=imperial_units
ELSEIF g_value=71 THEN ' metric mode
UNITS AXIS(0)=metric_units
UNITS AXIS(1)=metric_units
UNITS AXIS(2)=metric_units
ELSEIF g_value=90 THEN ' absolute mode
PRINT "Absolute Mode Set"
absolute=1
ELSEIF g_value=91 THEN ' relative mode
PRINT "Incremental Mode Set"
absolute=0
ELSE
PRINT "Ignoring unsupported token G";g_value[0]
ENDIF
RETURN

dispatch_m: ' machine code
IF m_value=2 THEN
PRINT "Program END"
ELSEIF m_value=3 THEN
PRINT "Spindle ON CW"
OP(10,ON) ' spindle on CW
OP(11,OFF)
ELSEIF m_value=4 THEN
PRINT "Spindle ON CCW"
OP(10,OFF)
OP(11,ON) ' spindle on CCW
ELSEIF m_value=5 THEN
PRINT "Spindle OFF"
OP(10,OFF) ' spindle off
OP(11,OFF)

```

```
ELSEIF m_value=6 THEN
  PRINT "Tool change"
  ' user's tool change routine here
ELSEIF m_value=7 THEN
  PRINT "Mist coolant ON"
  ' user's routine here
ELSEIF m_value=8 THEN
  PRINT "Flood coolant ON"
  ' user's routine here
ELSE
  PRINT "Ignoring unsupported token M";m_value[0]
ENDIF
RETURN

dispatch_s: ' spindle speed
  PRINT "Setting spindle speed: ";s_value[0]
  DAC AXIS(3)=s_value * 3.2768
  RETURN

dispatch_t: ' tool selection
  PRINT "Select Tool #";t_value[0]
  ' enter user's tool selection routines here
  RETURN

normalize:
  IF absolute THEN
    x = x_value - x_pos
    y = y_value - y_pos
    z = z_value - z_pos
    x_pos=x_value
    y_pos=y_value
    z_pos=z_value
  ELSE
    x = x_value
    y = y_value
    z = z_value
  ENDIF
  RETURN
```