QuickBuilder™

An Introduction to QuickBuilder Motion

Control Technology Corporation

Model 5300 Motion Programming Overview

Motion Architecture – The Big Picture

CTC's Model 5300 uses a powerful object-oriented approach to solve motion control applications. This greatly simplifies application creation and maintenance. It also improves performance by off loading the demanding motion control tasks to specialized motion control processors on the 5300 Motion Modules.

Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

Motion Architecture – The Big Picture

The main components used in Model 5300 motion control are:

1.The Axis Module The physical Motion Module in the rack

2.The Axis Object The QuickBuilder Resource representing an axis on that physical module

3.The MSB The **M**otion **S**equence **B**lock containing one or more motion statements that execute on the Axis Module's CPU under the supervision of **QuickStep™** on the main 5300 CPU

Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

The Axis Module

A 5300 axis module is inserted into the 5300 rack just like any other I/O module. CTC offers axis modules that can control one or more motion axes.

Each motion module contains its own CPU and Motion Accelerator Chip or MAC to ensure consistent high-performance motion control regardless of the number of axes to be controlled.



M3-40A Dual Servo Module I/O

Copyright © 2010 CTC

5300 Motion Programming — Overview

The Axis Module

There are currently three M3-40 modules that can be used in the Model 5300 automation controller:

- M3-40A: 2 Axis Servo Module
- M3-40B: 3 Axis Stepper/High-Speed Counter Module, 24V command
- M3-40C: 3 Axis Stepper/High-Speed Counter Module, 5V command



M3-40A Dual Servo Module I/O

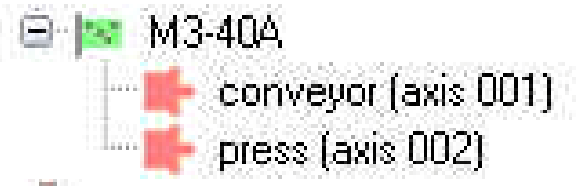
Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

The Axis Object

The Axis Object represents a physical servo or stepper axis on a motion module.



It is created automatically when a motion module is added to a rack in the Resource Manager.

Axis Objects have many specialized properties that can be configured using the Property Inspector.

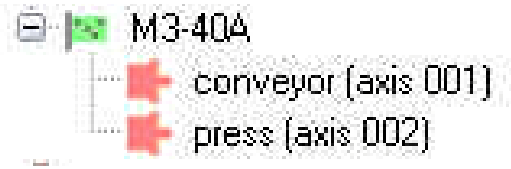
Most of these properties can also be changed dynamically in the QuickBuilder project.

Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

The Axis Object



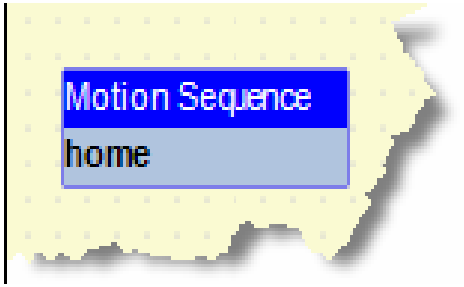
- Axis Objects have various inputs and outputs that control the servo (or stepper) and usually feedback signals that are used to monitor position.
- Using Motion Sequence Blocks (MSBs), you can command each axis to perform a sequence of motion statements.

Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

The Motion Sequence Block



- The Motion Sequence Block (MSB) element holds *QuickMotion* statement sequences.
- MSBs appear in the QuickBuilder project as stand-alone graphical elements.
- MSBs are not associated with any particular axis. This allows the same sequence to be reused many times for different axes, similar to the way a function works.

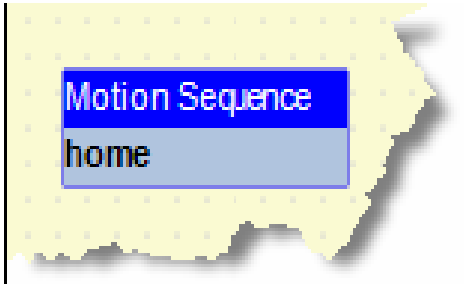
Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Programming — Overview

The Motion Sequence Block

- An MSB is started on a given axis in QuickStep™ by using the “Start MSB” statement.
- MSBs are programmed in the *QuickMotion* language – a language designed specifically for motion.
- One MSB can start another MSB that can run in parallel on the same axis:
  - Up to 4 foreground (500µs) MSBs can be running simultaneously.
  - Any number of background MSBs can be running simultaneously (memory limited).



Copyright © 2010 CTC

ctc Control Technology Corp.

QuickStep™ Motion Statements

There are only two Quickstep statements used to control motion on an axis in Quickstep4 (QS4):

start

- Begins execution of the named *Motion Sequence Block* (MSB) on the specified axis as a *background (BG) or foreground (FG)* MSB.

stop

- Stops execution of all foreground and background MSBs

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickStep™ Motion Statements

Start

- Begins execution of the named *Motion Sequence Block* (MSB) on the specified axis.
- This is a QS4 statement:

```
start axis1 EnableAxisBG;  //This is the statement to start a background
                           //MSB called Enable Axis on the axis named axis1
```

- This is a QS4 statement:

```
start axis1 EnableAxis;    //This is the statement to start a background
                           //MSB called Enable Axis on the axis named axis1
    (optional legacy form, where you don't have to specify BG)
start Xaxis  RegistrationFG;  //This is the statement to start a foreground
                           //MSB called Registration on the Axis called Xaxis
```

- It is *not* an error to start another MSB when there is one already running for a given axis – however, if the named MSB is already running on a given axis, the *start* is effectively ignored.

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickStep™ Motion Statements

Stop

Stops execution of foreground and background MSBs.

- **stop** axis1; //This is the statement to stop all MSBs on an axis from QS4

Copyright © 2010 CTC

ctc Control Technology Corp.

Understanding Modes

Operating Modes

POSITIONING

- In this mode, the axis is able to perform absolute and incremental time-based motion, including *SegmentedMoves* and time-based *CAMs*.
- The axis must have completed any pending positioning operations before changing to a different operating mode.

SLEWING

- In this mode, the axis generates a series of interpolated positions based on a constant (but alterable) velocity.
- The axis is put into slewing mode with the *slew begin* statement and returned to positioning mode by using the *slew end* statement.

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Programming — Overview

Operating Modes

TRACKING

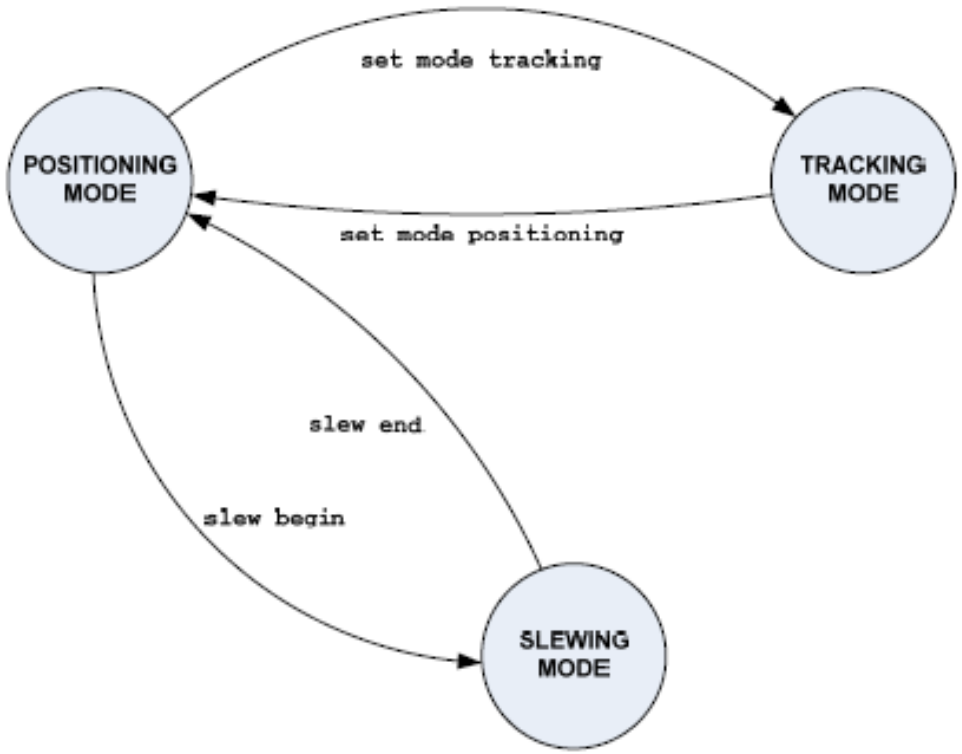
- The axis is able to perform position-tracking in this mode. This includes *following* and *gearing*.
- The axis must have completed all pending tracking operations before changing to a different operating mode.

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Programming — Overview

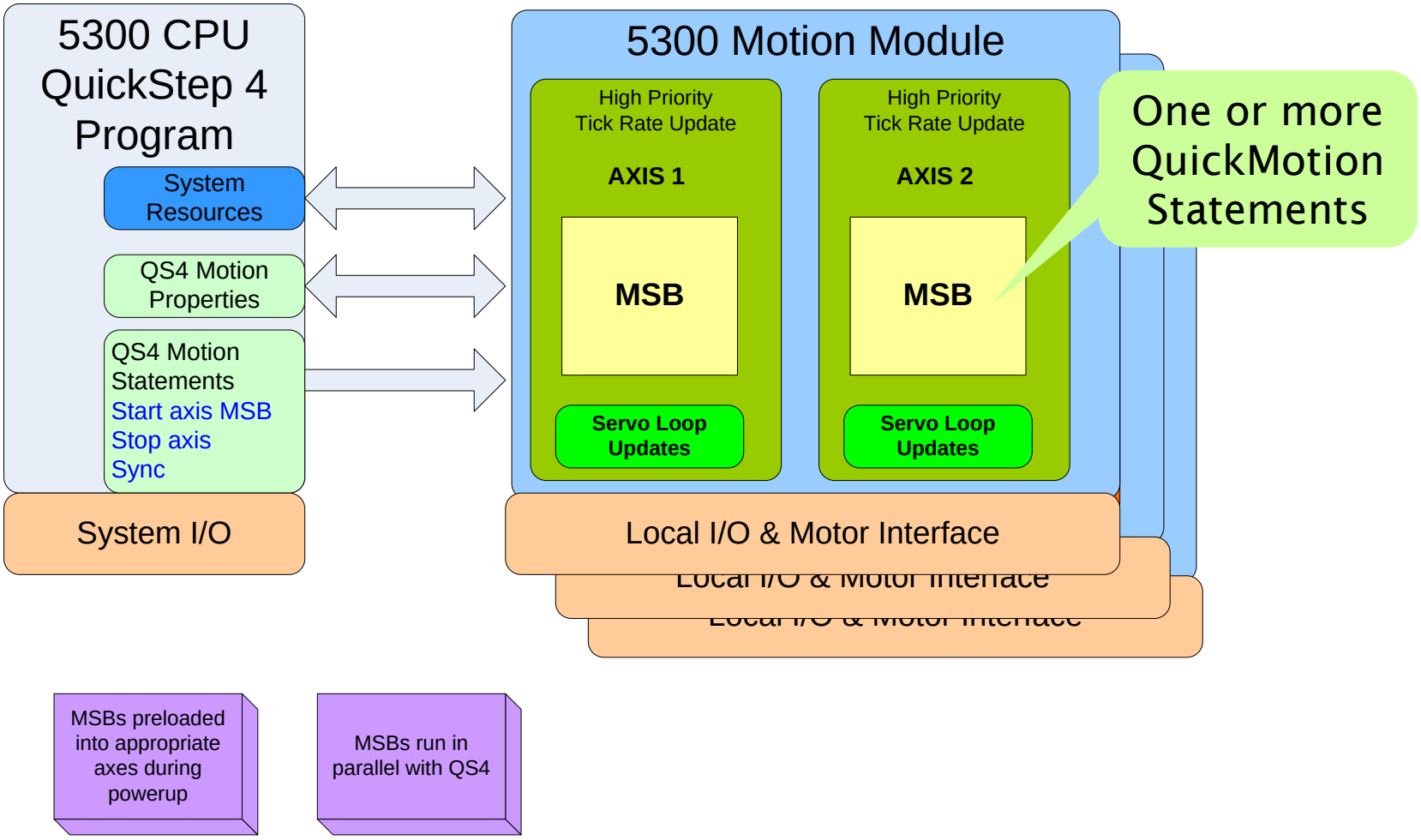
Operating Modes



Copyright © 2010 CTC

ctc Control Technology Corp.

5300 Motion Architecture Summary



Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Overview

This section covers statements that can be used within an MSB. These statements CANNOT be used in a QuickStep step!

- The MSB statement set was created to simplify the motion programming process and make powerful motion control applications accessible to a wide range of users.

- These statements are optimized for high-performance execution on the Model 5300 motion modules.

- In addition to the motion statement set, CTC provides over 100 pre-defined motion variables that can be used to simplify your application.

- Some important MSB variables are covered in a later slide.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Overview

The following slides review the main MSB statements, describing the functionality and giving usage examples.

QuickBuilder uses a specific color for each component of the MSB statement. For example:

delay 2500 MS; //delay for 2.5 seconds\

Explanation:

Blue = MSB statement Word

Black = literal (numeric value)

Red = variable or constant

Purple = CTC variable (e.g., ZPULSE_POS)

//Green = comment

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Label (label)

- A label within an MSB is used as a marker for the destination of a *goto* or similar statement.

- It is often necessary to *iterate* or *branch* depending on the state of an external input/output or internal condition. A label is used to mark the destination.

Statement Label	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax `[label]`				

Example: [Top] //this label is called Top

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Go to (goto)

- This statement causes program flow to jump/goto the specified label.

Goto	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`goto label`				
parameters				
`label`	the name of the label to branch to			

Example: goto Top; //jump to the MSB label called Top

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

If goto (if/goto)

- This statement causes program flow to jump/goto the specified label if a condition is true.
- If *false*, MSB program flow continues at the next MSB statement.

If/Goto	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
if condition goto label
```

parameters

`condition`	a Boolean test condition
`label`	the name of the label to branch to

Example: if din1 goto Makemove; //If the axis's input1 is on goto the label makemove

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Start (start)

This statement activates another MSB.

- If the MSB is already active, this statement is effectively ignored.
- Up to 4 foreground (**FG**) MSBs may be running simultaneously.
- There is no logical limit to the number of active background (**BG)** MSBs.

Start	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
start MSB mode
```

parameters

MSB	the name of the MSB to start	
mode	FG	start as a high-priority (tick) MSB
	BG	start as a low-priority (non-tick) MSB

Example:

start PressCap FG; //start the MSB called PressCap and run as a foreground MSB

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

End (end)

This statement ends execution of this MSB.

- An optional MSB can be specified to start after this one completes.
- An *end* or *goto* statement must be the last statement in any MSB.

End ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
end { and start MSB mode }
```

parameters

MSB	the name of the MSB to start	
mode	FG	start as a high-priority (tick) MSB
	BG	start as a low-priority (non-tick) MSB

Example 1:

```
end;                          //this is the end of the MSB
```

Example 2:

```
end and start WeldCap FG;     //end the current MSB and then start the MSB called
                              //WeldCap and run it as a foreground MSB
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements – Program Flow

Abort (abort)

- This statement ends execution of another MSB.
- If the named *MSB* is not active, the statement is effectively ignored.

Abort	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`abort MSB`				
parameters				
`MSB`	the name of the MSB to abort (stop) the execution of			

Example:

```
 if (perr >0.5 | perr < -0.5) then goto ABORT; //if there is too much err goto ABORT

[ABORT]
 abort WeldCap;  //kill the WeldCap MSB
stop;                  // stop the motor
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Stop (stop)

- In positioning mode:
  - *Non-slewed* – This statement immediately aborts the present motion operation and halts the target position generator from updating the target position (*tpos*) thereby (eventually) stopping motion.
  - *Slewed* – This statement first copies the current feedback position (*fpos*) into the target position (*tpos*) and then generates a controlled deceleration by *slewing* to zero velocity using the specified rate.
- If the axis is in *slewing* mode, a *slew end* is issued thereby placing the axis in *positioning* mode. The optional stop mode *slewed* is ignored in this mode.
- If the axis is in *tracking* mode, the numerator of the gear ratio is set to zero – but the axis remains in *tracking* mode. The optional stop mode *slewed* is ignored in this mode.

Stop ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
stop { slewed using rate }
```

parameters

rate	optional rate at which to stop the axis in user unit

Example: stop slewed using stoprate; //stop the axis at the rate specified

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Enable/Disable Drive
(drive enable) or (drive disable)

- Enables or disables the drive associated with the axis allowing motion to occur.
- Note: In some cases, the motor may slowly decelerate to zero velocity when disabling. This will be based on the drive/motor and how they are connected to the controller.

Enable/Disable Drive	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☐ FG MSB

syntax

```
drive enable

drive disable
```

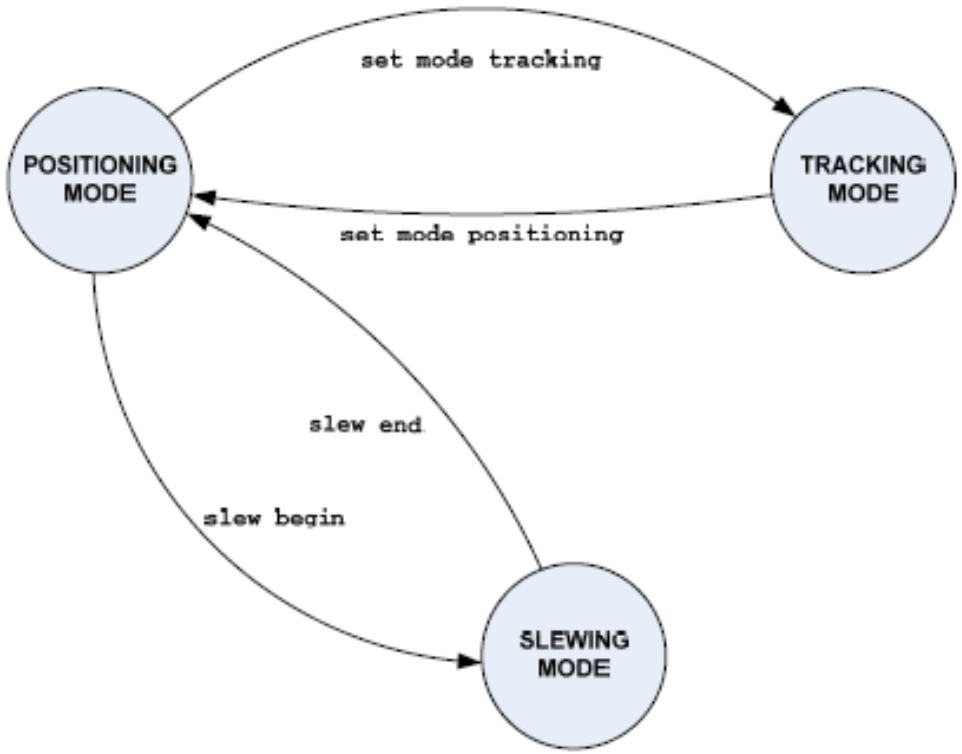
Example: drive enable; //enable the servo drive for this axis

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Programming — Operating Modes

Operating Modes



Copyright © 2010 CTC

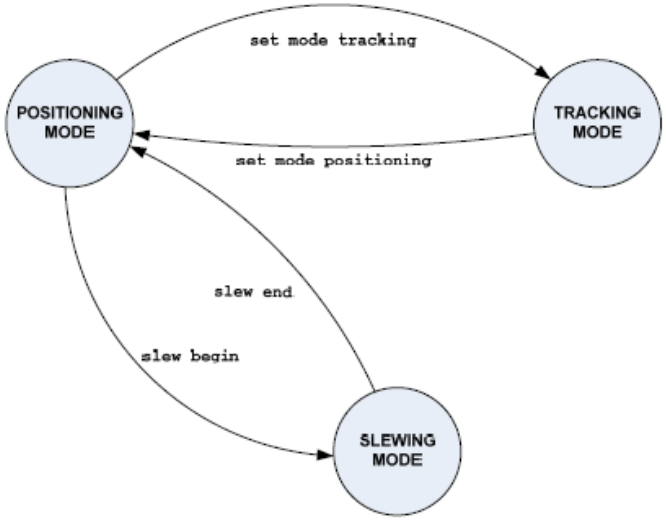
ctc Control Technology Corp.

MSB Statements — Utility Statements

Set Positioning Mode

(set mode positioning)

- Sets the operating mode of the axis to *positioning*.
- The axis must complete all pending tracking operations before changing to different operating mode.



Set Positioning Mode	☐ Positioning	☐ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
set mode positioning
```

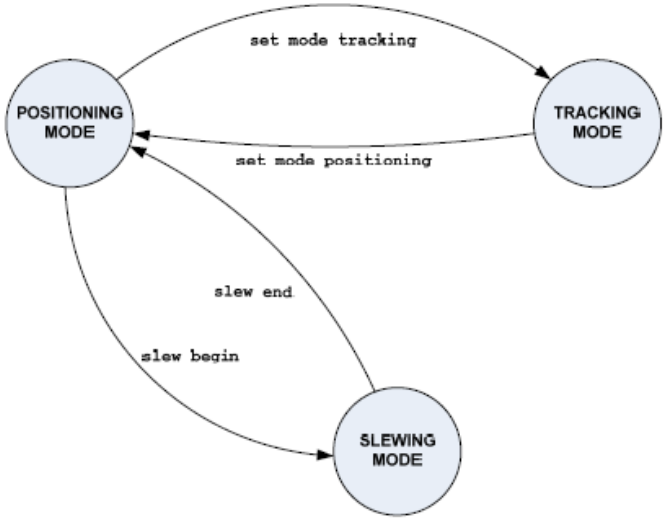
Example: set mode positioning; //switch to positioning mode

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Set Tracking Mode (set mode tracking)



- Sets the operating mode of the axis to *tracking (following)*.

Set Tracking Mode	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
set mode tracking
```

Example: set mode tracking; //switch to tracking mode so we can gear this axis

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Slew Begin (slew begin)

This statement changes the operating mode of the axis to *slewing (velocity mode)*.

Slew (begin)	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB
syntax `slew begin`				

Example:

```
slew begin;  //change from position mode to slew mode
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Slew End (slew end)

This statement changes the operating mode of the axis to positioning. A zero-speed slew (in 0.0 *time*) is first generated if the axis is currently slewing at a non-zero velocity.

Slew (end)	☐ Positioning	☑ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB
syntax `slew end`				

Example:
slew end; //stop motion is velocity is not zero and set mode to positioning mode

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Time delay (delay)

- This statement delays execution of the active MSB for the specified number of milliseconds.

Time Delay	☑Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`delay time ms`				
parameters				
`time`	an expression representing time in milliseconds			

Example: delay 2500 ms; //delay for 2.5 seconds
delay 2500; //optional legacy delay syntax

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Variable Assignment

- The value of the specified *expression* is evaluated and stored to the named *variable*.

Example:

```
host read Analogin, 8501;   // read analog input 1 from the CPU and assign it to Analogin
Speed = 34.857 * Analogin;  //calculates a value for speed based on analogin
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Zero Feedback Position

(zero feedback position)

- Zeros the feedback position, but maintains following error.

Zero Feedback Position	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
zero feedback position
```

Example: zero feedback position; //set the current position as zero

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Zero Target Position

(zero target position)

- Zeros the target position, but maintains following error. Operates the same as *zero feedback position*, but is more readable in stepper mode.

① Zero Target Position	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
zero target position
```

Example: zero target position; //set the current position as zero

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Zero Feedback/Following Error (zero following error)

- Zeros the feedback position and target position, thereby removing any following error.
  - Useful in torque applications (e.g., nut driving)

Zero Following Error	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
zero following error
```

Example: zero following error; //"relax" the system by zeroing the following error

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Set/Offset Target/Feedback Position (set/offset/target/feedback position)

- set target position & set feedback position set the position to a specific absolute value in user units.
- set target position counts & set feedback position counts set the position to a specific absolute value in counts.
- offset position & offset position counts modify the target and feedback positions simultaneously by adding the specified offset to *both*.

Note: Following error is maintained when these statements are executed.

The axis must not be *active (*actively generating a target position by use of a *move* statement) when any of these statements are executed.

Set/Offset Position(s)	Target/Feedback	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
set target position value
set feedback position value
set target position counts vcounts
set feedback position counts vcounts
offset position value
offset position counts vcounts
```

parameters

value	new or offset for the named position (user-units)
vcounts	new or offset for the named position (counts)

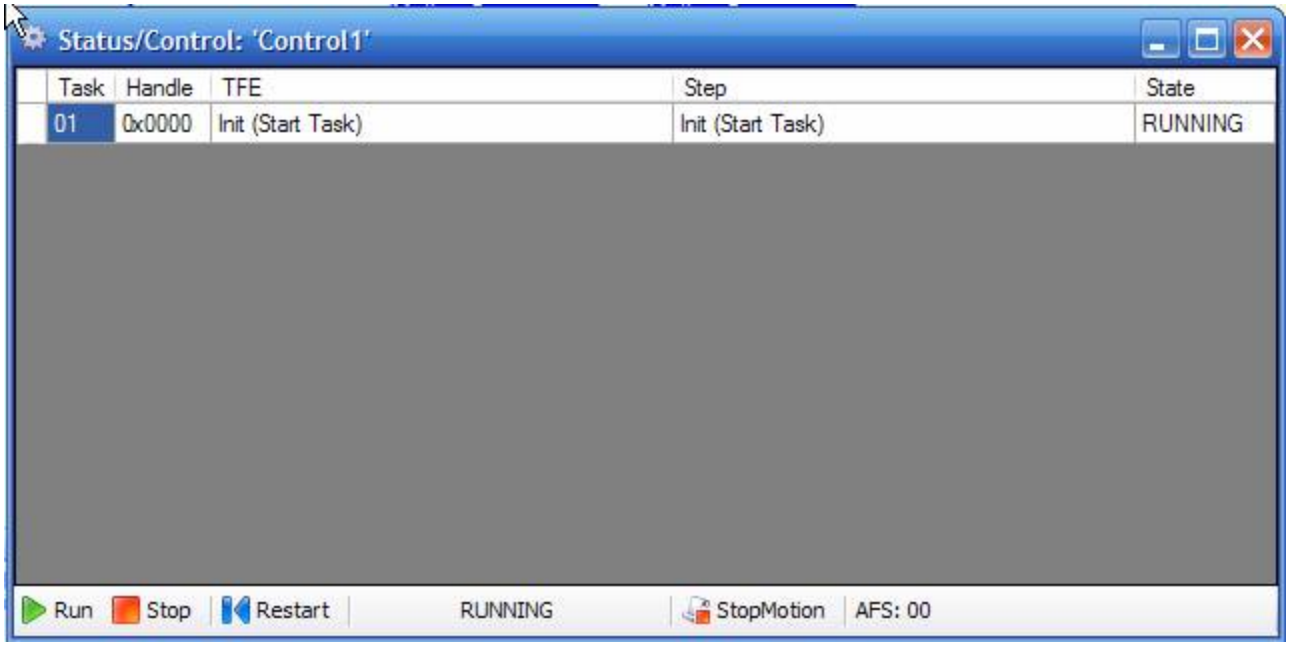
Example:
offset position counts 1100;
// offsets both the target and feedback positions by 1100 counts

Copyright © 2010 CTC

ctc Control Technology Corp.

Monitoring Motion Faults

You can check the fault status of any axis in the Control Status Window:



AFS:00 means that both axes do not have any faults.



The red AFS:F0 means there is a fault on the first axis.

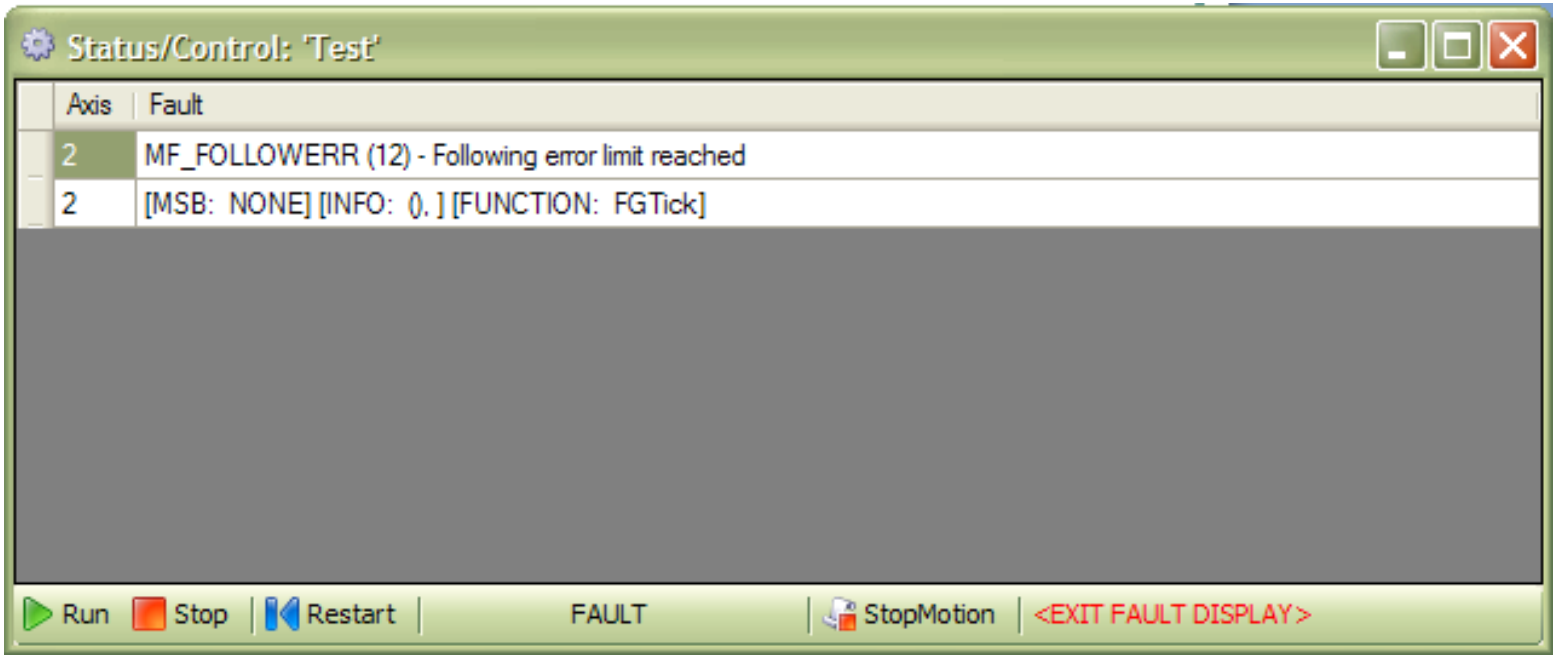
Copyright © 2010 CTC

ctc Control Technology Corp.

Monitoring Motion Faults



Clicking on the AFS (Axis fault Status) bring up any fault messages:



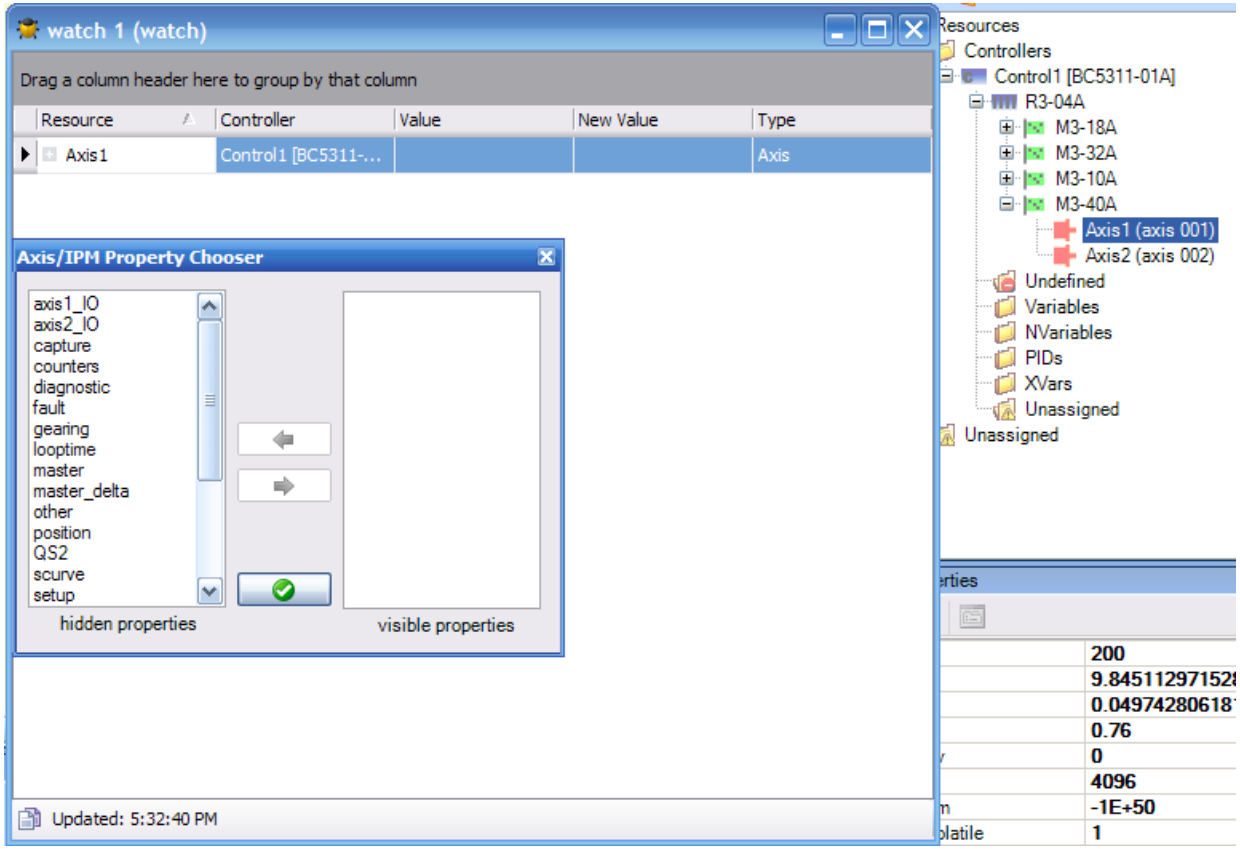
Clicking on <EXIT FAULT DISPLAY> gets you back to the standard Status/Control Window.

Copyright © 2010 CTC

ctc Control Technology Corp.

Monitoring Motion — Watch Window

You can drag an axis from the Resource Window into the Watch Window:



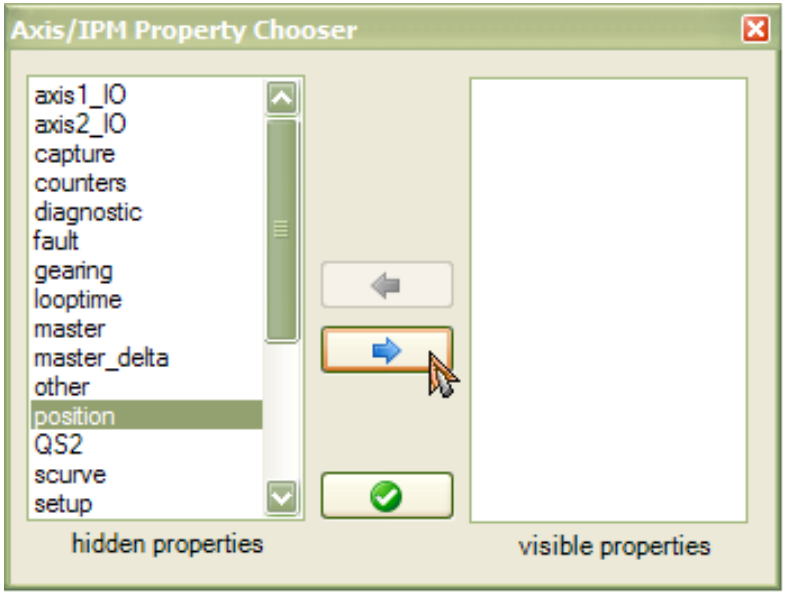
Copyright © 2010 CTC

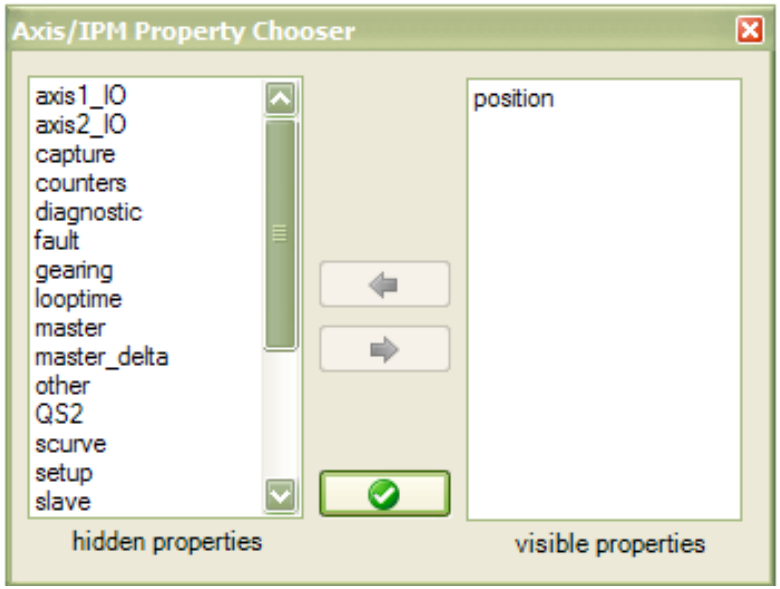
ctc Control Technology Corp.

Monitoring Motion — Watch Window

The Axis/IPM Property Chooser allows you to select which Motion properties to monitor for that axis.

Simply select the item(s) you want to monitor from the list of hidden properties and click on the add arrow to add them to the list of visible properties. Then click the green check box.





Copyright © 2010 CTC

ctc Control Technology Corp.

Monitoring Motion — Watch Window

This is then the resulting information you will see in the Watch window when selection position for one of your axis.

Resource	Controller	Value	New Value	Type
Axis2	Test [BC5311-01A] : R3-04A (1)			Axis

Resource Table_Property Table

Name	Value	NewValue	Type
tpos	-0.0439453125		ipo
tposc	-1		ipo
fpos	-2.5048828125		ipo
fposc	-57		ipo
sfposc	-56		ipo
sfmod	0		ipo
perr	2.4169921875		ipo
ztpos	0		ipo
zfpos	0		ipo
mpos	0		ipo
mposc	0		ipo
sfpos	-0.0069580078125		ipo
activeCAM_row	0		ipo
dac_mv	0		ipo
ipos	-0.0001220703125		ipo

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Lab 1: Tuning

In Motion Lab 1, you will create a simple back-and-forth move to introduce the use of MSBs and Servo Tuning.

See ***Motion Lab 1*** in the Motion Lab Manual.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

If /Assignment (if / then)

- This statement evaluates the specified *condition*.
  - If *true*, the *expression* is evaluated and the *variable* is set to the resulting value.
  - If *false*, MSB program flow continues at the next MSB statement.

If/Assignment	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
if condition then variable = expression
```

parameters

condition	a Boolean test condition
variable	a variable
expression	an expression

Example:

```
if perr > .25 then Fault = 2;      // if the position error for Axis1 exceeds
                                   //0.25 set a fault code to 2
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Utility Statements

Wait Until (wait until)

- This statement waits until the specified *condition* is *true*.

Wait Until	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`wait until condition`				
parameters				
`condition`	condition to test			

Example: wait until Temp > 32.849; //wait here until chamber temp exceeds min value

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Resource Access

Host Read

Pauses execution of the MSB while the contents of a QuickBuilder Register are retrieved from the main processor

- The register value is stored in the local “variable” or axis “property”.
- The data type is automatically be converted to that of the local storage.
- Integer-based registers, variant vectors, and tables are supported.
- When reading a variant, one cell at a time in the table (if any) is read.
- If no row or column is specified, 0 is assumed.

The next slide shows the syntax and an example of a host read statement.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Resource Access

Host Read **(host read)**

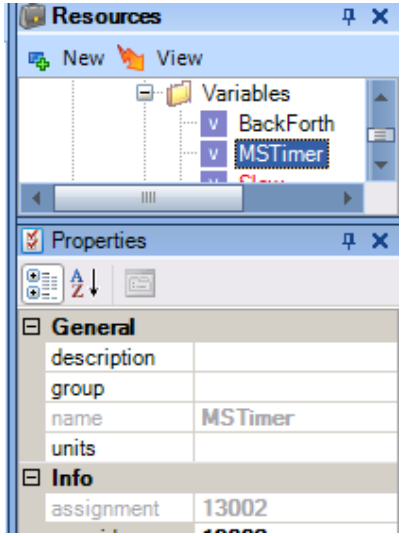
Host Read ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
host read variable, register {, row, column }
```

parameters

variable	Local user variable or axis property to have 'register' stored to it.
register	Main controller QuickBuilder register as defined in the Model 5300 Quick Reference Register Guide. May be constant or variable access.
row	Optional row used only for variant register table access. May be constant or variable access.
column	Optional column used only for variant register table access. May be constant or variable access.



Example:

```
// Read the controller tick timer referencing a variable
// and store to 'userVar'
Reg = 13002;
host read userVar, Reg;
// Read the controller tick timer using constant register
// number and store to 'userVar'
host read userVar, 13002;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Resource Access

Host Write

Pauses execution of the MSB while a local “variable” or “axis property” is written to a QuickBuilder register on the main processor.

- The data type isautomatically be converted to that of the QuickBuilder register, thus double will be converted to integer, etc.
- Integer-based registers, variant vectors, and tables are supported.
- When reading a variant, one cell at a time in the table (if any) is read.
- If no row or column is specified, 0 is assumed.

The next slide shows the syntax and an example of a host write statement.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Resource Access

Host Write (host write)

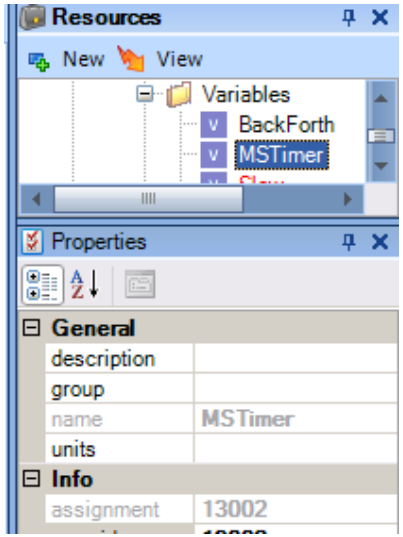
Host Write ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
host write variable, register {, row, column }
```

parameters

variable	Local user variable or axis property to store to controller 'register'.
register	Main controller QuickBuilder register as defined in the Model 5300 Quick Reference Register Guide. May be constant or variable access.
row	Optional row used only for variant register table access. May be constant or variable access.
column	Optional column used only for variant register table access. May be constant or variable access.



Example:

```
// Clear the controller tick timer referencing register
// 13002 referencing the contents of 'userVar'
UserVal = 0;
Reg = 13002;
host write userVal, Reg;
// Clear the controller tick timer referencing register
// 13002 using a constant value.
host write userVal, 13002;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Lab 2 — Resource Usage

In Motion Lab 2, you will modify Lab 1 to introduce the access of Quickstep4 Resources.

See ***Lab 2*** in the Motion Lab Manual.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Variables

User Variables

48 User Variables per axis

- double precision float
- No need to define them
- Accessible from both the MSBs for THAT axis and QS4.

QS4 example code:

```
Axis1.x = DrillPosition;        // set the variable X for Axis1 to DrillPosition
Axis1.Speed = 5;                // set the variable speed for Axis1 to 5
start Axis1 MSB1;               //start MSB1 on axis1
```

Axis1 MSB example code:

```
Halfspd = Speed/2;
move at Halfspd to x;           /*Make a trap move to the DrillPosition specified
                                in the QS4 step at half speed  */
wait for inposition;             //Wait for move to complete
pulse 1 for 1000;      // Turn the drill output on for 1 sec
move at Speed to Home;          // move to home at the speed specified above.
```

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Lab 2: Variables

In Motion Lab 2a, you will modify Lab 2 to introduce the use of variables.

See ***Lab 2a*** in the Motion Lab Manual.

Copyright © 2010 CTC

ctc Control Technology Corp.

Common Bits and Variables

Common bits and common vars are used to communicate state information and are **much** faster then standard variables:

a. between QuickMotion based modules
b. between QuickMotion and QuickStep 4
c. between axes on a single module such as an M3-40A

- There are 256 common bits and 256 common vars. Common bits are Boolean, and common vars are bytes and, therefore, have values from 0 through 255.
- The common bits are shared globally among all QuickMotion modules as well as QuickStep 4. Any changes made to common bits are “seen” by all QuickMotion modules and the main CPU running QuickStep 4.
- The first 32 common vars are *overlaid* on top of the 256 common bits – changes made to a common var may alter up to 8 common bits.
- The remaining 224 common vars are *module-local* – changes are only seen local to the module. This is useful to communicate state information between axes on a two-axis QuickMotion module such as an M3-40A.

Copyright © 2010 CTC

ctc Control Technology Corp.

Common Bits and Variables

When referring to common bits in QS4 the $CBITS vector system variable is used.

- The $CBITS vector is a Boolean read/write system variable with 256 elements ([0] to [255])
- $CBITS[0] is used to access common bit 0.

Examples:

```
// wait for CBITS[4] to be on
while !$CBITS[4] repeat { }

// set CBITS[2] on
$CBITS[2] = 1;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

Common Bits and Variables

When referring to common vars in QS4 the $CVARS vector system variable is used.

- The $ CVARS vector is a read/write system variable taking values of 0 through 255 with 32 elements ([0] to [31]).
- $CVARS[0] is used to access common var 0.

Examples:

```
// wait for CVARS[22] to be > 20
while $CVARS[22] <= 20 repeat { }

// set CVARS[2] to 11
$CVARS[2] = 11;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Variables

Set Common Bit (set common bit)

Set Common Bit statement sets a common bit to a specified state.

Set Common Bit	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
set common bit number state
```

parameters

number	bit number (0-255)
state	true or false

```
set common bit 45 true;  //set common bit 45 to 1
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Variables

Wait for Common Bit (wait for common bit)

Wait for Common Bit statement waits for a common bit to be set to a specified state.

Wait For Common Bit	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
wait common bit number state
```

parameters

number	bit number (0-255)
state	true or false

Example AXIS 2 Shuttle MSB Code:

```
move at HalfSpd to DrillPosition; //Make a trap move to DrillPosition at HalfSpd
wait for in position;             //Wait for move to complete
set common bit 5 true;            //change the common bit 5 to on
wait for common bit 5 false;
move at Speed to Home;
```

Example AXIS 1 Drill MSB Code:

```
wait for common bit 5 true;
move at HalfSpd to 3.72;   //Make a trap move to 3.72 inches from home atHalfSpd
wait for in position;      //delay execution of MSB until move is complete
move at Speed to 0;        //Make a trap move to home
wait for in position;      //delay execution of MSB until move is complete
set common bit 5 false;    //set the common bit 5 to off
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Variables

Set Common Var (set common var)

This statement sets the specified “common state variable” is to a specified value.

Set Common Var	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
set common var number value
```

parameters

number	variable number (0-255)
value	an integer value (0-255)

```
[ErrorCheckLoop]  //check to see if there is too much position error
if (perr >= 200 || perr <= -200) goto Positionerr;
if !din2 goto LightCurtain;  //check for the lightcurtain input
goto ErrorCheckLoop;
[Positionerr] // if too much position error set cvar0 to 1
set common var 0 1;
goto ErrorCheckLoop;
[LightCurtain]  // if Light Curtain is tripped set cvar0 to 2
set common var 0 2;
goto ErrorCheckLoop;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Variables

Wait for Common Var (wait common var)

Wait for Common Var statement waits for a “common var” to be set to a specified value.

Wait For Common Var	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
wait common var number range
```

parameters

number	variable number (0-255)	
range	x	a value of x
	x-y	a value of x through y inclusive
	! x	a value other than x
	! x-y	a value outside the range of x through y

AXIS 1 MSB Example Code:

Using the code from the previous slide for axis 1 we can then have another axis check common var 0 before allowing motion on that axis.

AXIS 2 MSB Example Code:

```
wait common var 0 0;  //waits for the common var 0 to be at 0
move at 20 for 15;    //now that the other axis is ready move this one
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Set Output(s) (setout)

- Sets one or more outputs to *on*.
- The output number can be 1-5 (dual-axis mode).

Clear Output(s)	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`setout outputlist`				
parameters				
`outputlist`	a comma delimited list of outputs to set			

Examples:

```
setout 2;          //turns on the second output on the servo card
setout 1, 3;       //turns on the first and third output on the servo card
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Clear Output(s) (clrout)

- Sets one or more outputs to *off*.
- The output number can be 1-5 (dual-axis mode).

Clear Output(s)	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
clrout outputlist
```

parameters

outputlist	a comma delimited list of outputs to clear

Examples:

```
clrout 2;          //turns off the second output on the servo card
clrout 1, 3;       //turns off the first and third output on the servo card
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Pulse Output (pulse n for n)

This statement causes the specified *output* to pulse for the specified duration.

- If the output is already *on* when this statement executes, the output state is unchanged – however it will be turned *off* after the specified time.
- If another statement changes the state of the *output* (for example, to *on* or *off*) before the allotted duration, the generation of the pulse is aborted.

Pulse Output	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`pulse output for n`				
parameters				
`output`	the output to pulse 1-5 (dual axis mode) 1-10 (1½ axis mode)			
`n`	the time to pulse the output, an expression as milliseconds			

Example: pulse 2 for 500; /*turns on the second output on the servo card for 500ms */

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Wait For Input (wait for rise of n)

This statement waits for the specified *transition* of the specified general purpose *input* to occur.

- The MSB will not continue execution until the transition occurs – unless there was a *condition* specified and the condition evaluated to *true*.
- When this statement is used with the optional exit *condition* and the statement is part of a BG MSB, it is possible to miss transitions of the general purpose input. Therefore, the optional exit *condition* form should be used with care in BG MSBs.

Example: wait for rise of 1;
//delay execution of MSB until input1 transitions from off to on

Wait For Input	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
wait for transition of input { or condition }
```

parameters

`transition`	`rise or fall`
`input`	the general purpose input to wait upon 1-5 (dual axis mode) 1-10 (1½ axis mode)
`condition`	an optional exit condition

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Lab 3: I/O

In Motion Lab 3, you will modify Lab 2 to introduce the use of simple I/O.

See ***Lab 3*** in the Motion Lab Manual.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Generate Pulses (generate output n rate n)

This statement begins or ends generation of pulses using a specific output.

If pulses are being generated on an output, then *setout*, *clrout,* and *pulse output* commands given to the same output have the following result:

- ***setout*** **-** no pulse generation occurs; the output will be active
- ***clrout*** **-** pulse generation occurs for non-zero *freq*s
- ***pulse output*** **-** no pulse generation occurs until the *pulse output* completes

When a frequency of 0 is specified, no pulse generation occurs.
This effectively turns the output back into a general-purpose output.

The *minimum* frequency that can be generated is 1 Hz. The *maximum* frequency that can be generated is well over 500 kHz.

Example:

```
generate output 2 rate 500;
/* generate 500Hz pulse on
output 2 */
```

Generate Pulses	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
generate output output rate freq
```

parameters

output	1-10
freq	the frequency (in Hz) to generate pulses; rounded to an integer

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Generate Steps (generate n steps on axis n)

This statement allows control of a stepper motor using step and direction outputs.

- Generates step and direction pulses on the specified step and direction pair.
- If the expression *n* evaluates to a negative number, then the direction will be negative.
- Any *setout*, *pulse* or *generate output* used in parallel with this command will cause erroneous step/dir pulses to be emitted. You should not used these commands in conjunction with *generate steps*.

Generate Steps	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
generate n steps on pair
```

parameters

`n`	the number of steps to generate in 500 µsec
`pair`	the step/direction pair to output steps on: 1. axis 1 step/direction pair (M3-40A/B/C outputs 3&4) 2. axis 2 step/direction pair (M3-40A/B/C outputs 3&4) 3. alternate step/direction pair (outputs 5 on each axis)

Example: generate 1000 steps on 1; //send the stepper drive 1000 pulses using axis 1 outputs 3 & 4

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Alternate Stepper Output
(generate alternate on/generate alternate off)

Alternate Stepper Output	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
generate alternate mode
```

parameters

mode	**on**	generate stepper outputs on alternate pins
	off	generate stepper outputs on standard pins

- On a M3-40A/B/C, (in stepper mode in the case of the M3-40A), the step and direction outputs are normally output on axis (TBx) pin pairs (15, 16).
- These cards also allow a third axis to be controlled by temporarily outputting step and direction pulses on TB1 pin 22 (step) and TB2 pin 22 (direction).
- To output on this alternate pair, issue the command *generate alternate on*. To output on the standard pair, issue the command *generate alternate off*.
- **Be careful** – only the destination of the step and direction signals change – to the axis that motion is being commanded on the primary axis (and thus updates it updates its idea of where the absolute stepper position is). Therefore, it is good practice to zero the target position ("*zero target position*") before switching to or from this alternate mode.

Copyright © 2010 CTC

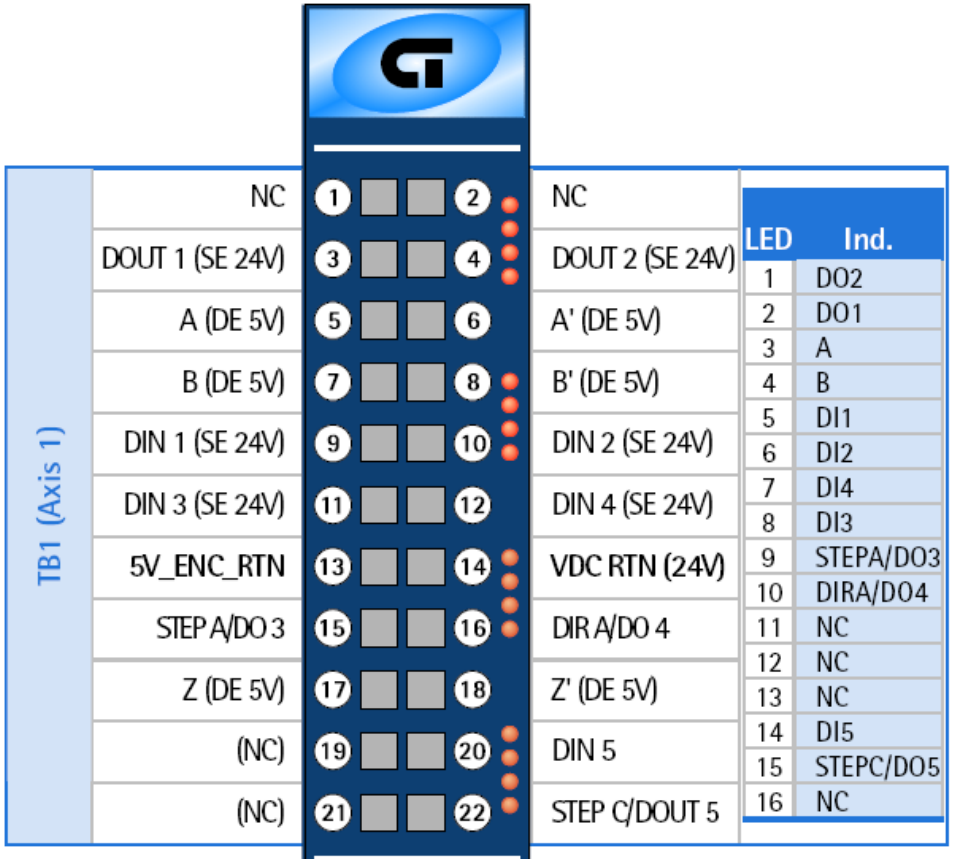
ctc Control Technology Corp.

MSB Statements — I/O Statements

Alternate Stepper Output

Example:

```
// move my axis 30 revs
zero target position;
generate alternate off;
move at 5 for 30 using 10,10;
wait for in position;
// move axis #3 20 revs
wait common bit 1 true;   //wait for Y move
zero target position;
generate alternate on;
move at 10 for 20 using 10,10;
wait for in position;
// move me again 10 revs
zero target position;
generate alternate off;
move at 5 for 10 using 10,10;
wait for in position;
```



LED	Ind.
1	DO2
2	DO1
3	A
4	B
5	DI1
6	DI2
7	DI4
8	DI3
9	STEPA/DO3
10	DIRA/DO4
11	NC
12	NC
13	NC
14	DI5
15	STEPC/DO5
16	NC

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

PLS Define ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
pls output using reference definitions
```

parameters

output	the output (1-5) to control via a PLS
reference	the encoder count scaled reference variable to compare to:
	fposc — Feedback position of axis msb
	mposc1 - mposc5 — Master position counters #1 through #5
	mposc — Master position counter
	smodc — Slave position (modulo)
	smark — Slave marked position
	tmc1 tmc2 — Temporary master counters #1 & #2
	tsc1 tsc2 — Temporary slave counters #1 & #2
	sdc — Slave decrement counter
	fposc1 — Feedback position of axis 1 (fposcA)
	fposc2 — Feedback position of axis 2 (fposcB)
	tmodc — Temporary master counter mod mmc
	sfposc — Secondary feedback position of axis
definitions	a comma-separated list of up to 16 PLS definitions:
	on x to y — Turn output on when the reference is within the bounds specified by x through y (may be expressions)

PLS Define

This statement defines or redefines a PLS (software-based programmable limit switch) associated with a given output.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

PLS Define (*continued*)

This statement defines or redefines a PLS (software-based programmable limit switch) associated with a given output.

- A definition overwrites the previous definition for an output (if one was defined already)...
- When a PLS is defined/re-defined. it will be disabled and will not compute the state for the output.
- To enable a PLS after it is defined/re-defined, a *pls on* statement must be issued.

Copyright © 2010 CTC

MSB Statements — I/O Statements

PLS Enable/Disable **(pls using** and **pls on)**

This statement enables (“on”) or disables (“off”) a PLS for an output.

- If the output is *on* when a PLS is disabled, it will remain *on* – unless you re-enable the PLS (to re-compute the PLS output), or you *clrout* the output.

PLS Enable/Disable	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax				
`pls output state`				
parameters				
`output`	the output to control via a PLS 1-5 (dual axis mode) 1-10 (1½ axis mode)			
`state`	`on or off`			

Example: //define a PLS for output #1
pls 1 **using** fposc **on** 10 **to** 200, **on** 400 **to** 430;
/*output will be on when fposc is within 10-200 or 400-430*/
pls 1 **on;** // enable the PLS for output #1

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Counter Read Write/Offset

These specialized forms of the *assignment* statement give read/write/offset access to the axis counters.

On the M3-40A, -40B and -40C, there are 8 counters/axes that accumulate off-to-on transitions of the following:

ctr[0]	digital input #1
ctr[1]	digital input #2
ctr[2]	digital input #3
ctr[3]	digital input #4
ctr[4]	digital input #5
ctr[5]	‘A’ channel input (non-quadrature)
ctr[6]	‘B’ channel input (non-quadrature)
ctr[7]	‘Z’ channel input (non-quadrature)

Copyright © 2010 CTC

MSB Statements — I/O Statements

Counter Read Write/Offset

These specialized forms of the *assignment* statement give read/write/offset access to the axis counters.

- The first form of the statement stores the current counter value in a variable.
- The second form of the statement changes the current counter value.
- The third form of the statement offsets the current counter value using +/- =.

Example:

```
[top]
// wait until input #1 rises
wait for rise of 1;
 // get the current count value
x = ctr[7];
 // accumulate
Totalcounts = Totalcounts + x ;
// offset so no counts are missed
ctr[7] -= x;
goto top;
```

Counter read, write, offset	☑ Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB

syntax
`variable = ctr[n]`
`ctr[n] = expression`
`ctr[n] = offset`				

parameters				
`n`	the counter number (0 through 7)			
`variable`	the variable to store the current value of the counter to			
`expression`	a new value for the counter			
`offset`	an offset for the counter (subtracted from the current counter value)			

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Asynchronous Event Handling ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
on asynchevent asynchhandler
```

parameters

asynchevent	One of the following:	
	riseof *n*	Rise of specified general purpose input.
	fallof *n*	Fall of specified general purpose input.
	hardfault	When a non-recoverable fault occurs.
	capture	Capture of specified input trigger.
	pls *output*	PLS output 1 to 5 activated.
	timeout	'timerticks' variable decrements to 0 (use 'set timeout' to initialize msb private value).
asynchhandler	One of the following:	
	ignore	Cancel asynchronous event monitoring.
	start *MSB {FG/BG}{arm}*	Starts the specified MSB in BG (background) mode unless FG is specified, if capture then optional {arm} at end of statement.
	goto *label {arm}*	Branch on event, if capture then optional {arm} at end of statement.

Asynchronous Event Handler

This statement controls asynchronous event handling.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — I/O Statements

Asynchronous Event Handler

This statement controls asynchronous event handling.

- If *asynchhandler* is set to *start*, an MSB is started automatically when the specified event occurs. If the *MSB* is already active when the event occurs, a second instance is *not* started. Ending an MSB stops the Asynch event.
- If *asynchhandler* is set to goto, a branch to that label occurs upon the event, within the same MSB.
- If *asynchhandler* is set to cancel, no operation will occur upon event. Each event is unique to a specific MSB although only one MSB may monitor a capture or specific pls output event.

Example: on riseof 3 start WeldCap BG; //wait rising edge of DIN3; then start WeldCap

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Asynchronous Event Handling ☑ Positioning ☑ Slewing ☑ Tracking ☑ BG MSB ☑ FG MSB

syntax

on *asynchevent asynchhandler*

parameters

asynchevent	One of the following:	
	riseof *n*	Rise of specified general purpose input.
	fallof *n*	Fall of specified general purpose input.
	hardfault	When a non-recoverable fault occurs.
	capture	Capture of specified input trigger.
	pls *output*	PLS output 1 to 5 activated.
	timeout	'timerticks' variable decrements to 0 (use 'set timeout' to initialize msb private value).
asynchhandler	One of the following:	
	ignore	Cancel asynchronous event monitoring.
	start *MSB {FG/BG}{arm}*	Starts the specified MSB in BG (background) mode unless FG is specified, if capture then optional {arm} at end of statement.
	goto *label {arm}*	Branch on event, if capture then optional {arm} at end of statement.

Asynchronous Event Handler

This statement controls asynchronous event handling.

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Asynchronous Event Handler

This statement controls asynchronous event handling.

- If *asynchhandler* is set to *start*, an MSB is started automatically when the specified event occurs. If the *MSB* is already active when the event occurs, a second instance is *not* started. Ending an MSB stops the Asynch event.
- If *asynchhandler* is set to goto, a branch to that label occurs upon the event, within the same MSB.
- If *asynchhandler* is set to cancel, no operation will occur upon event. Each event is unique to a specific MSB although only one MSB may monitor a capture or specific pls output event.
- If *asynchevent* is set to timeout, the 'set timeout <ticks>' command must be set for down counting to begin (500uS/tick). Branching based upon a timeout will occur regardless of motor operations. It is up to the MSB to properly recover and/or stop motors.

Example: on hardfault goto Error; //If a hardfault occurs goto the Error label in this MSB

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Program Flow

Timeout Initialization (set timeout)

This command initializes a private MSB timer that is decremented on every tick if the 'on timeout' command is active.

- To disable, execute an 'on timeout ignore' command.
- The timeout value must be set after every timeout; it acts as a down counter, invoking the event handler when 0 is reached.

Timeout Initialization	☑Positioning	☑ Slewing	☑ Tracking	☑ BG MSB ☑ FG MSB
syntax `set timeout ticks` *parameters*				
`ticks`	Number of ticks until a timeout occurs causing any active 'on timeout' event handlers to take action.			

Example:

```
x = 0;
y = 0;
set timeout 5000 * 2;   //5 sec timeout
on timeout goto timedout;
[top]   //x will increment for 5 seconds and then a branch to [timedout] will occur
x = x + 1;
delay 100 ms;
goto top;
[timedout]  //y will increment after 5 sec and continue forever
y = y+1;
delay 100 ms;
goto timedout;
```

Copyright © 2010 CTC

MSB Statements — Simple Motion

We will now cover simple motion statements including both absolute and incremental moves.

- Absolute and incremental moves will appear very similar.
  - Absolute moves will “move to” a specific position.
  - Incremental moves will “move for” a specific distance.

- All of the move commands can use either literals (numeric values) or variable to define the motion distances, speeds, and accelerations.

- S-Curve support is optionally available for the move commands, from a stopped position.
  - jerk_a_req and jerk_d_req are typically set to 1 when S-curve accerlations are needed

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Absolute, Triangular (move to)

This statement generates a triangular move to the specified end *position*.

- If the parameters *acc* and *dec* are omitted, then the default rates are used.
- Linear acceleration and deceleration is used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.
- The specified *position* may also be specified as ZPULSE_POS or ZPULSE_NEG meaning the next encoder Z-pulse in the positive or negative directions, respectively.

Example1: move to Drillpos; /*move to the absolute position specified by the variable drillpos using default acceleration and deceleration rates. */

Example2: move to ZPULSE_POS; /*move in the positive direction until the Z pulse is seen using default acceleration and deceleration rates. */

Move Absolute, Triangular	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☐ FG MSB

syntax

```
move to position { using acc, dec }
```

parameters

`position`	absolute end position, user-units
`acc`	acceleration rate, user-units/sec/sec
`dec`	deceleration rate, user-units/sec/sec

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Absolute, Speed-limited (move at n to n)

Generates a trapezoidal move to the specified end *position*.

- If it is not possible to reach the specified maximum velocity *maxvelocity*, then a triangular move is generated.
- If the parameters *acc* and *dec* are omitted, then the default rates are used.
- Linear acceleration and deceleration is used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.
- The specified *position* may also be specified as ZPULSE_POS or ZPULSE_NEG meaning the next encoder Z-pulse in the positive or negative directions, respectively.

Example:

```
move at Rapidrate to Drillpos;
/*move to the absolute position
specified by the variable
drillpos using default
acceleration and deceleration
rates and the rapidrate variable
for a max velocity. */
```

Move Absolute, Speed-limited ☑ Positioning ☐ Slewing ☐ Tracking ☑ BG MSB ☐ FG MSB

syntax

```
move at maxvelocity to position { using acc, dec }
```

parameters

`maxvelocity`	unsigned maximum velocity, user-units/sec
`position`	absolute end position, user-units
`acc`	acceleration rate, user-units/sec/sec
`dec`	deceleration rate, user-units/sec/sec

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Absolute, Trapezoidal (move trap to n using n)

This statement generates a ⅓-⅓-⅓ trapezoidal move (⅓ of the time accelerating, ⅓ constant velocity, ⅓ decelerating) to the specified end *position*.

- The acceleration and deceleration *rate* must be specified.
- Linear acceleration and deceleration is used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.
- The specified *position* may also be specified as ZPULSE_POS or ZPULSE_NEG meaning the next encoder Z-pulse in the positive or negative directions, respectively.

Example:

```
move trap to Drillpos using Rapidacc;
/*move to the absolute position
specified by the variable drillpos using
the variable rapidacc to set acceleration
and deceleration rates. The velocity
used will be based on the calculation
to achieve a trap move*/
```

Move Absolute, Trapezoidal ☑ Positioning ☐ Slewing ☐ Tracking ☑ BG MSB ☐ FG MSB

syntax

```
move trap to position using rate
```

parameters

`position`	absolute end position, user-units
`rate`	acceleration/deceleration rate, user-units/sec/sec

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Absolute, Time-limited (move in n to n)

This statement generates a ⅓-⅓-⅓ trapezoidal move to the specified end *position* in the specified *time*.

- Linear acceleration and deceleration are used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value, in which case an S-curve-type profile is generated.
- The specified *position* may also be specified as ZPULSE_POS or ZPULSE_NEG meaning the next encoder Z-pulse in the positive or negative directions, respectively.

Move Absolute, Time-limited	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
move in time to position {mode n }
```

parameters

time	time, sec
position	absolute end position, user-units
n	acc / dec ramp multiplier

Example:

```
move in Movetime to Drillpos;  /*move to the absolute position specified by the
variable drillpos setting the calculated velocity, accel and decel rates to make a
trapezoidal move in the time specified by the variable movetime. */
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Incremental, Triangular (move for n)

This statement generates a triangular move for a specified *displacement*.

- If the parameters *acc* and *dec* are omitted, then the default rates are used.
- Linear acceleration and deceleration are used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.

Example:

```
move for Spanmove;
/*move an incremental
distance specified by the
variable spanmove using
default  acceleration and
deceleration rates. */
```

Move Incremental, Triangular ☑ Positioning ☐ Slewing ☐ Tracking ☑ BG MSB ☐ FG MSB

syntax

```
move for displacement { using acc, dec }
```

parameters

displacement	incremental position, user-units
acc	acceleration rate, user-units/sec/sec
dec	deceleration rate, user-units/sec/sec

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Incremental, Speed-limited (move at n for n)

This statement generates a trapezoidal move for a specified *displacement*.

- If it is not possible to reach the specified maximum velocity *maxvelocity*, a triangular move is generated.
- If the parameters *acc* and *dec* are omitted, the default rates are used.
- Linear acceleration and deceleration is used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value, in which case an S-curve-type profile is generated.

Example:

```
move at Slowspeed for Spanmove;
/*move an incremental distance
specified by the variable spanmove
using default  acceleration and
deceleration rates and using the
variable slowspeed as a max velocity.
*/
```

Move Incremental, Speed-limited ☑ Positioning ☐ Slewing ☐ Tracking ☑ BG MSB ☐ FG MSB

syntax

```
move at maxvelocity for displacement { using acc, dec }
```

parameters

`maxvelocity`	unsigned maximum velocity, user-units/sec
`displacement`	incremental position, user-units
`acc`	acceleration rate, user-units/sec/sec
`dec`	deceleration rate, user-units/sec/sec

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Incremental, Trapezoidal (move trap for n using n)

This statement generates a ⅓-⅓-⅓ trapezoidal move (⅓ of the time accelerating, ⅓ constant velocity, ⅓ decelerating) for a specified displacement.

- The acceleration and deceleration *rate* must be specified.
- Linear acceleration and deceleration are used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.

Move Incremental, Trapezoidal	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☐ FG MSB

syntax

```
move trap for displacement using rate
```

parameters

displacement	incremental position, user-units
rate	acceleration/deceleration rate, user-units/sec/sec

Example:

```
move trap for Offset using Rapidacc;  /*move the incremental distance specified by
the variable offset using the variable rapidacc to set acceleration and deceleration
rates. The velocity used will be based on the calculation to achieve a trap move*/
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Move Incremental, Time-limited

This statement generates a ⅓-⅓-⅓ trapezoidal move for a specified displacement in the specified *time*.

- Linear acceleration and deceleration are used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.

Example:

```
move in Movetime for Offset;
/*move the incremental distance specified by the variable offset setting the calculated velocity, accel and decel rates to make a trapezoidal move in the time specified by the variable movetime. */
```

Move Incremental, Time-limited ☑ Positioning ☐ Slewing ☐ Tracking ☑ BG MSB ☑ FG MSB

syntax

```
move in time for displacement {mode n }
```

parameters

time	time, sec
displacement	incremental position, user-units
n	acc / dec ramp multiplier

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Timebase (speed override)

Timebase is actually a predefined variable for any given axis and not a command statement.

- Used to override the natural time base of the target generator. When set to 1.0 (the default value), the target generator's "time" is un-scaled.
- When set to a value between 0.0 and 1.0, the target generator's "time" is slowed-down, effectively generating lower velocities. When set to 0.0 motion stops (allowing you to pause motion).
- *Changing the timebase only effects commanded motions, it does not alter other commands such as delay.*

Example:

```
move in 2.0 for Offset;  //move an incremental distance
[loop]
if din1 then timebase = .5;      // if input1 is on run at half speed/accel/decel
if din2 then timebase = 1.0;     // if input 2 is on run at full speed/accel/decel
goto loop;
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Set New End Position (new endposition relative n using n)

This statement modifies the end position for the active *move* command. If there is no active *move*, this statement is effectively ignored.

- The first form of this statement changes the end position to a new *absolute* position.
- The second form of this statement changes the end position *relative to the current position*.
- Using a *displacement* of 0 effectively stops motion *here* without generating a fault (unlike the *stop* command).
- A *rate* must be specified. R*ate* is used as the acceleration/deceleration rate for the modified profile.
- **DSP** Linear acceleration and deceleration are used (as programmed in the axis *acc* and *dec* properties) unless the property *jerk* is set to a non-zero value in which case an S-curve type profile is generated.
- **ZPULSE_POS** or **ZPULSE_NEG** should not be used with this motion command.

Set New End Position	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
new endposition position using rate
new endposition relative displacement using rate
```

Example:

```
move at 5 to 25;     // start moving to 25 mm
wait for rise of 1;  //when din1 is activated during the move,
new endposition relative 3 using 10; /*change the end position of the move to 3
                                       from when the input seen */
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Wait for In Position (wait for in position)

This statement temporarily stops the execution of the active MSB until the target generator has reached its final value and the position error, *perr* is within the programmed in-position window.

Wait For In Position	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
wait for in position
```

Example:

```
move at Slowspeed for Spanmove;  /*move speed-limited
wait for in position;  //wait till motor is within the programmed in-position window
pulse 1 for 1000; //Turn on output 1 for 1 second
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Slew Begin (slew begin)

This statement changes the operating mode of the axis to *slewing (velocity mode)*.

Slew (begin)	☑ Positioning	☐ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
slew begin
```

Example:

```
move at Slowspeed for Spanmove;  /*move speed-limited
wait for in position;  //wait till motor is within the programmed in-position window
slew begin;  //change from position mode to slew mode
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Slew At (slew at n in n)

This statement alters the current slew *velocity*. The velocity is changed smoothly over the specified *time*. For an immediate speed change, specify 0.0 for *time*.

Slew At	☐ Positioning	☑ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
slew at velocity in time
```

parameters

velocity	new slew velocity, user-units/sec
time	time, sec

Example:

```
slew begin;  //change from position mode to slew mode
slew at Feedrate in 0.5;  //change from current speed to feedrate in 0.5 seconds
```

Copyright © 2010 CTC

ctc Control Technology Corp.

MSB Statements — Simple Motion

Slew For (slew for)

This statement alters the current slew velocity over time (to a slew velocity of 0.0) such that some *displacement* is consumed.

- If the current slew velocity is 0.0, this statement is ignored.
- The *displacement* should be unsigned, as the sign of the current slew velocity is used to sign the *displacement*.

Slew For	☐ Positioning	☑ Slewing	☐ Tracking	☑ BG MSB ☑ FG MSB

syntax

```
slew for displacement
```

parameters

displacement	ending relative slew position, user-units

Example:

```
slew begin;  //change from position mode to slew mode
slew at Feedrate in 2;  //change from current speed to feedrate in 2 seconds
wait for rise of 3;  //delay execution of MSB until input3 transitions from off to on
slew for Registrationoffset;  /*slew to a stop in the distance specified by the variable
                                registrationoffset */
```

Copyright © 2010 CTC

ctc Control Technology Corp.

Motion Lab 3: Homing

The Homing Lab introduces the concept of a “Master” MSB to control other MSBs. The choice to use a “Master” MSB vs. starting MSBs from QS4 is a personal one. You may find that using a combination of a “Master” MSB to start some MSBs and starting a few MSBs from QS4 may work best for you.

This Lab also introduces the use of the On-board inputs and slewing to accomplish a few homing methods.

See ***Motion Lab 3*** in the Motion Lab Manual.

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

In addition to user-defined variables, there are a number of pre-defined variables for an axis in the QuickMotion language. Many of these variables correspond to properties in the QS4 world.

No more keeping track of cryptic register settings!

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Status Variables

enabled

Holds the state of *drive enable*.

inpos

Holds the state of *in position*.

In Position is *true* when the target generator is *inactive* and when the position error (*perr*) is within bounds set by *inposw*.

overpos

Set to *true* when feedback position (*fpos*) exceeds *poslim*. Software limit.

overneg

Set to *true* when feedback position (*fpos*) exceeds *neglim*. Software limit.

overtrq

Set to *true* when the statemented torque *trqc* has been clamped to the torque limit (*tlim*).

rmstrq

The RMS of the statemented torque (Nm).

time

A settable, accurate (within 500usec) time counter (sec).

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Control Variables – fixed on start-up

cmode

Control mode – Determines the command signal the controller sends out. Set to [Torque], Velocity or Stepper.

This is set using the *property inspector* (and thus cannot be changed in QM code) for the axis to one of the following values:

- **Torque** – Control loop outputs a torque statement (velocity loop is active).
- **Velocity** – Control loop outputs a velocity statement to the drive (velocity loop is inactive).
- **Stepper** - Control loop outputs step and direction pulses to the drive (velocity loop is inactive).

tmax

Scale factor – maximum torque (Nm) that is generation at the motor when the control loop statements 10V to the drive.

- This is set using the *property inspector* and cannot be changed in QM code.
- This property is valid when *cmode* is *Torque*.

vmax

Scale factor – velocity generated when then control loop statements 10V to the drive. Scaled in RPM (rotational) or linear-units/min (linear).

- This is set using the *property inspector* and cannot be changed in QM code.
- This property is valid when *cmode* is *Velocity*.

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Control Variables – dynamically changeable

tlim

Torque limit (Nm) – torque statement limit

acc

Default acceleration rate for absolute and incremental motion
Scaled in user-units/sec/sec

dec

Default deceleration rate for absolute and incremental motion
Scaled in user-units/sec/sec.

Copyright © 2010 CTC

QuickMotion Pre-defined Variables

Control Variables – dynamically changeable

jerk_a_req

requested acceleration jerk in units/sec3, read/write. Set to 1 for automatic calculation.

jerk_d_req

requested deceleration jerk in units/sec3, read/write. Set to 1 for automatic calculation.

jerk_a

acceleration actual jerk, read only

jerk_d

deceleration actual jerk, read only

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Control Variables – dynamically changeable

timebase

Used to override the natural time base of the target generator.

When set to 1.0 (the default value), the target generator's "time" is un-scaled. When set to a value between 0.0 and 1.0, the target generator's "time" is slowed-down, effectively generating lower velocities.

gtimebase

A global timebase variable that affects both axes. This variable in conjunction with the per-axis *timebase* sets the effective per-axis natural time base of the target generator.

This parameter should only be set through a reference to the first axis on each card in an MSB or from a QS4 program.

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Tuning Variables

kvf

Velocity-loop factor (0.0-1.0).

- When set to 1.0, the velocity loop is a classic PID structure.
- When set to 0.0, the velocity loop is a classic PDF structure.
- When set to a value in between, the velocity loop is a combination of both.

kv

Velocity-loop proportional gain (P).

- Scaled as Nm/RPM or Nm/(in r/s or linear-units/sec).

ki - Velocity-loop integral gain (I).

- Scaled as Nm/RPM/sec or Nm/(in r/s or linear-units/sec)/sec.

kd - Velocity-loop derivative gain (D).

- Scaled as Nm-sec/RPM or Nm-sec/(in r/s or linear-units/sec).

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Tuning Variables

aff

Velocity-loop acceleration feed-forward gain.

- Scaled as Nm/RPM/sec or Nm/(in r/s or linear-units/sec)/sec

vff

Velocity-loop velocity feed-forward gain.

- Scaled as Nm/RPM or Nm/(in r/s or linear-units/sec)

ppg

Position-loop proportional gain

- Scaled as 1000/min

pff

Position-loop velocity feed-forward gain (0.0 – 1.0)

pdead

Position-loop dead-band (user-units)

Copyright © 2010 CTC

QuickMotion Pre-defined Variables

Feedback Variables

ppr

Feedback encoder counts per revolution (or per *linear unit* for linear feedback devices).

sppr

Steps/rev to output when in stepper mode (cmode=stepper)

mppr

Master encoder counts per revolution (or per *linear unit* for linear feedback devices).

uun

User-units conversion factor (*numerator*).

- Motion statements are multiplied by this value (then divided by uud) to scale user-units to revolutions (or *linear unit* or linear feedback devices).

uud

User-units conversion factor (*denominator*).

- Motion statements are divided by this value (after multiplying by uun) to scale user-units to revolutions (or *linear unit* or linear feedback devices).

tpos

The target position scaled in user-units

tposc

The target position scaled in encoder counts

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Feedback Variables

fpos

The feedback position scaled in user-units.

fposc

The feedback position scaled in encoder counts.

sfposc

A secondary feedback position (scaled in counts).

- A separately maintained feedback position similar to *fposc* with the exception that the position will “wrap” (modulo) at 0 and at *sfmod* (unless *sfmod* is set to 0).

sfmod

The secondary position modulus.

- Used to control when *sfposc* wraps around to 0.

perr

The position error (scaled in user-units).

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Feedback Variables

trqc
The statemented torque value (Nm).

vcmd
statemented velocity (in r/s or linear-units/sec).

vel
Feedback velocity (in r/s or linear-units/sec).

verr
Velocity error (in r/s or linear-units/sec).

inposw
In Position window - Controls when the axis is deemed *in position*. Scaled in user-units.

poslim
Positive over-travel limit, scaled in user-units.

neglim
Negative over-travel limit, scaled in user-units

Copyright © 2010 CTC

QuickMotion Pre-defined Variables

Feedback Variables

ztpos

Holds the last target position before it was modified by a "*zero feedback position*" or "*zero following error*" statement

zfpos

Holds the last feedback position before it was modified by a "*zero feedback position*" or "*zero following error*" statement

invertfeed

Whether to invert the way the feedback encoder counts:

0 = count normally

1 = count inverted

invertmaster

Whether to invert the way the master encoder counts:

0 = count normally

1 = count inverted

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Axis I/O Variables

din1 – din5
The state of digital inputs 1 through 5

dout1 – dout5
The state of digital outputs 1 through 5

dins
The state of digital inputs 1 through 5 as a single integer

douts
The state of digital outputs 1 through 5 as a single integer

Copyright © 2010 CTC

ctc Control Technology Corp.

QuickMotion Pre-defined Variables

Axis I/O Variables

reg1 – reg2

The state of registration inputs 1 and 2

overposin

The digital input number to use for positive over-travel

0 = disable positive over-travel checking

overnegin

The digital input number to use for negative over-travel

0 = disable negative over-travel checking

driveenable

The digital output number to use for “drive enable”

0 = use no output

running

The digital output number to use for “an MSB active” (running)

0 = use no output

Copyright © 2010 CTC

ctc Control Technology Corp.