

Following—Statements

These statements are designed to be used with the Model 4000-CFM Option.

FOL

Name	FOL					
Descriptor	Following Parameters					
Type	Set-Up					
Default	N/A					
Syntax	FOL					
Options	TAB	MASTER	SHIFT	ENABLE	RATIO	ETC
	TAB	MDIST	MOVEWT	NEWCYC	MAS_CYC	ETC
	TAB	WAIT	SMOOTH	MAXACC	MAXVEL	ETC
	TAB	VELFF	CYC_OFF	M_SYNC	S_SYNC	ETC
	TAB	SYNC_OFF	WIN_P	WIN_W	PTOL	ETC
	TAB	LEAD	DIRSET	ENCCHK	CAM	ETC
		F 1	F 2	F 3	F 4	F 5

Description

FOL statements can be used only if you have purchased the Model 4000's Following option. FOL statements define master and slave parameters when one or more axes is involved in following an encoder or step output signal. Below is a summary of the Following statements.

FOL MASTER	Specify following master input.
FOL SHIFT	Execute time-based moves upon ratio following moves
FOL ENABLE	Enable and disable following mode.
FOL RATIO	Establish maximum allowed ratio for preset moves, or final ratio for continuous moves.
FOL MDIST	Set master distance for subsequent MOVE statements. MDIST specifies master distance over which preset moves take place, or master distance over which continuous moves change from one ratio to another.
FOL MOVEWT	Next move wait for trigger or master cycle position.
FOL NEWCYC	Define the start of a new master cycle, i.e., set the master cycle position to 0.
FOL MAS_CYC	Define master cycle length.
FOL WAIT	Wait for trigger or master cycle position.
FOL SMOOTH	Specify velocity measurement smoothing.
FOL MAXACC	Define following maximum acceleration
FOL MAXVEL	Define following maximum velocity
FOL VELFF	Enable or disable velocity feed forward
FOL CYC_OFF	Define initial master position
FOL M_SYNC	Define master synchronization mark
FOL S_SYNC	Define slave synchronization mark
FOL SYNC_OFF	Define expected synchronization position difference
FOL WIN_P	Define master window position
FOL WIN_W	Define master window width
FOL PTOL	Define following error tolerance
FOL CAM	Enable or disable cam profiling
DEFINE TRIGDB	Sets debounce time for trigger inputs
FOL DIRSET	Enables or disables direction change setup time
FOL ENCCHK	Enable or disable encoder/motor step check
FOL LEAD	Advance setpoint proportional to slave speed

See Also: FOLM, UNIT MASTER

FOL MASTER

Name	FOL MASTER					
Descriptor	Assign Master to Slave					
Type	Set-Up					
Initial value	None					
Range	± MOT or ENC, 1-4					
Default	FOL MASTER * * * *					
Syntax	FOL MASTER ENC2 MOT4 NO -ENC4					
Options	TAB	ENC	NULL	MOT	NO	
	F1	F2	F3	F4	F5	F6

Description

The **FOL MASTER** statement configures an axis to be a slave, but *does not* automatically enable following. To enable following use the **FOL ENABLE YES** statement. Any incremental or absolute encoder input, or any motor step output can be used as the master for any axis. As soon as the master is specified with the **FOL MASTER** statement, a continuously updated relationship between the position of the slave and the position of the specified master is maintained. The slave motor resolution and velocity ranges are used in these calculations. If the slave is in encoder step positioning (i.e., **MODE E_ABS** or **MODE E_INC**) then the encoder resolution of the slave axis is also used. For that reason, *these parameters may not be changed after the FOL MASTER statement configures an axis as a slave.* **FOL MASTER NO** releases an axis from a slave configuration, and returns it to normal operation.

Notice that the master input axis number does not need to be the same as the slave axis number. Axis 1 uses the encoder input on axis #2 as the master, axis #2 is a slave to the step output of axis #4, axis #3 is not configured as a slave, and axis 4 is a slave to the encoder input of that axis. If a slave axis is in encoder mode (**MODE E_INC** or **E_ABS**), or if stall detect or position maintenance is enabled, that axis can not use its own encoder input as the master. Also, a slave can not use its own motor step output as the master input. On power-up, and at the end of every program, no axis is configured as a slave.

A minus sign is allowed as a parameter for the **FOL MASTER** when describing the encoder or motor step master. The minus sign will be needed for applications in which the desired direction of positive master motion results in negative counts on the master. This is particularly true for preset moves as described below. The master can be the motor step output or encoder step input of any axis. Putting a minus sign in front of the master parameter specification in the **FOL MASTER** statement causes the incoming count to be negated before it is used by the slave. The term *master count* refers to the count after negation, if any.

For preset slave moves, the direction the slave travels depends on the mode of operation (absolute or incremental) and the commanded position. However, once a preset slave move is commanded, it will only start moving if the master is counting up. This is true no matter the commanded direction of the slave move.

For continuous slave moves, the master count direction has a different affect. If the commanded move is positive in direction, **MOVE SLEWCW**, and the master is counting up, the actual slave travel direction will be positive. If the commanded move is positive in direction, **MOVE SLEWCW**, and the master is counting down, the actual slave travel direction will be negative. Similar cases exist for slave moves commanded in the negative direction.

Each of the statements described below indicates the initial value taken as a result of the FOL MASTER statement

The **FOL MASTER** statement re-initializes all **FOL** and **FOLM** parameters each time it is executed. More information about preset and continuous slave moves can be found in the *Technical Considerations* section of this chapter.

See Also: **FOL**, **FOLM**

FOL SHIFT

Name	FOL SHIFT					
Descriptor	Following Phase Shift Move					
Type	Motion					
Initial value	None					
Range	±99999999 steps after scaling					
Default	FOL SHIFT * * * *					
Syntax	FOL SHIFT 12500 CW Q1 -525.67					
Options	TAB	Q	NULL	CW	CCW	ETC
	TAB	STOP	KILL			ETC
	F1	F2	F3	F4	F5	F6

Description

The FOL SHIFT statement allows time-based slave moves to be super-imposed on continuous following moves. Continuous shift moves in the CW or CCW direction, as well as preset shift moves of defined or variable distances may be commanded while a slave is performing a SLEWCW or SLEWCCW ratio move at any constant ratio. The velocity and direction of the SHIFT is independently super-imposed on whatever velocity and direction results from the ratio and motion of the master. The SHIFT is not a change in ratio. It is a velocity added to a ratio. Distances are scaled by UNIT POS. The FOL SHIFT parameters STOP and KILL can be used to halt a continuous or preset FOL SHIFT move (CW or CCW). The example below shows how to stop a FOL SHIFT continuous move. It should be noted that FOL SHIFT is similar in execution to MOVE and not MOVI. The entire preset distance shift or ramp to shift velocity must finish before the Model 4000 proceeds to the next statement. As with MOVE, however, a FOL SHIFT statement may be interrupted by an ON condition becoming true.

The most recently commanded VEL and ACCEL for the slave axis will determine the speed at which the FOL SHIFT move takes place. The velocity commanded will be added to the present speed at which the slave is moving, up to the velocity limit defined with the FOL MAXVEL statement. For example, assume a slave is traveling at 1 rps in the positive direction while following a master. If a FOL SHIFT move is commanded in the positive direction at 2 rps, the slave's actual velocity (after acceleration) will be 3 rps, assuming that FOL MAXVEL is greater than 3 rps.

A FOL SHIFT move may be needed to adjust slave position on the fly because of some load condition which changes during the continuous following move. For example, suppose an operator is visually inspecting the slave's motion with respect to the master. If they notice that the master and slave are out of synchronization, it may be desirable to have an interrupt programmed that will allow the operator to move the slave at a super-imposed correction speed until the operator chooses to have the slave start tracking the master again. The example below illustrates this.

See Also: FOL RATIO, FOL ENABLE

Example

Assume all scale factors and set-up parameters have been entered for the master and slave. In the example below, the slave (axis #1) is continually following the master at a 1:1 ratio. If the operator notices some mis-alignment between master and slave, there is 1 of 2 pushbuttons he can press to move the slave in the CW or CCW direction until the button is released. After the adjustment, the program continues on as before.

<u>Statement</u>	<u>Description</u>
PATT1 XXXX ... XX10	'Define input pattern #1
PATT2 XXXX ... XX01	'Define input pattern #2
ON IN24 = PATT1 GOTO SHIFT+	'Interrupt to shift slave in the CW direction when pattern 1 active
ON IN24 = PATT2 GOTO SHIFT-	'Interrupt to shift slave in the CCW direction when pattern 2 active
FOL MASTER ENC 4 * * *	'Axis 4 Encoder input is the master
FOL RATIO 1:1 * * *	'Set slave to master following ratio
FOL ENABLE YES * * *	'Enable following mode on axis #1
MOVE SLEWCW * * *	'Start following master continually
LABEL MAINLOOP	'Main program loop

```

      .
      'All other program operation takes place
      'within this loop
GOTO MAINLOOP      'Repeat main program loop
DONE               'End of program
LABEL SHIFT+      'Subroutine to shift in the CW 'direction
FOL SHIFT CW * * * 'Start slave shift in CW directions
WAIT FOR BIT2 = 0 'Continue shift until bit2 is 'deactivated
FOL SHIFT STOP * * * 'Stop shift move
ON IN24 = PATT1 GOTO SHIFT+ 'Re-enable interrupt for future shifts
GOTO MAINLOOP     'Return to main program loop
LABEL SHIFT-      'Subroutine to shift in the CCW 'direction
FOL SHIFT CCW * * * 'Start slave shift in the CCW 'direction
WAIT FOR BIT1 = 0 'Continue shift until bit #1 is
                  'deactivated
FOL SHIFT STOP * * * 'Stop shift move
ON IN24 = PATT2 GOTO SHIFT- 'Re-enable interrupt for future shifts
GOTO MAINLOOP     'Return to main program loop

```

FOL ENABLE

Name	FOL ENABLE					
Descriptor	Enable or Disable Following					
Type	Programming					
Initial value	NO					
Default	FOL ENABLE * * * *					
Syntax	FOL ENABLE NO * YES YES					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The **FOL ENABLE** statement indicates whether subsequent moves will be following a master (**FOL ENABLE YES**) or normal time-based moves (**FOL ENABLE NO**). The term *Following mode* simply means that **FOL ENABLE YES** has been given, and that the motion of the slave is dependent on the motion of the master at all times. If **FOL ENABLE NO** is given the motion of the master is still monitored but the motion of the slave is independent of the master. In order to enable following mode, the master must have been previously specified with the **FOL MASTER** statement. The **FOL ENABLE** statement may be used in the standard stationary positioning system, or in the moving positioning system.

See Also: **FOL RATIO**, **FOL MDIST**

FOL RATIO

Name	FOL RATIO					
Descriptor	Slave to Master Following Ratio					
Type	Set-Up					
Initial value	Ø					
Range	Maximum of 127 slave steps per master step					
Default	FOL RATIO * * * *					
Syntax	FOL RATIO 10:1.4 Q5:1 * 1					
Options	TAB	Q	NULL	:		
	F1	F2	F3	F4	F5	F6

Description

The **FOL RATIO** statement establishes the ratio between slave and master speed and position in terms of user units. For a preset move, it is the maximum allowed ratio, and for a continuous move, it is the final ratio reached by the slave. The ratio can be specified either with standard decimal numbers, or numeric **Q** variables. **FOL RATIO** is specified with two positive numbers, but it applies to moves in both directions. Actual slave direction will depend on commanded moves and master direction.

Assume **FOL RATIO** is set to 0.5:0.3 for an axis. The first parameter is scaled by the **UNIT POS** value to give slave steps. The second parameter is scaled by the **UNIT MASTER** value to give master steps. For a **UNIT POS** of 25000 and a **UNIT MASTER** of 4000, the slave to master step ratio would be 0.5*25000 to 0.3*4000, or 125 slave steps for every 12 master steps. If no second parameter is specified, it is assumed to be 1. Numeric **Q** variables can be used with this statement for slave and/or master parameters. The Model

4000 divides the scaled numerator and denominator to calculate the ratio. After scaling, the maximum value is 127 slave steps for every master step.

See Also: FOL ENABLE, UNIT MASTER, UNIT POS

Example

In the statements below, assume the master has a 1000 line incremental encoder on the back of a motor and programming units are to be rps. This yields a post quadrature UNIT MASTER scale factor of 4000. The motor resolution of the slave axis is 25000 steps/rev. A slave UNIT POS scale factor of 25000 provides consistent user units.

The slave will start ramping to a ratio of 1:1 when trigger #1 goes active. This means the actual step ratio of slave to master is 25000 to 4000, or 6.25 slave steps for every master. After 25 master revolutions, the slave will decelerate to a 0.5:1 ratio (3.125 slave steps for every master). After a total of 75 master revolutions, the slave will stop and repeat the cycle on trigger #1.

Statement	Description
UNIT POS 25000 * * *	'Set slave scale factor to 25000 steps/rev
UNIT MASTER 4000 * * *	'Set master scale factor to 4000 steps/rev
FOL MASTER ENCL * * *	'Assign encoder input #1 as master for axis #1
FOL MDIST 1 * * *	'Slave should accelerate over 1 master revolution
FOL MAS_CYC 100 * * *	'Set master cycle length to 100 revs
FOL RATIO 1:1 * * *	'Initial slave to master ratio is 1 to 1
FOL ENABLE YES * * *	'Enable following on axis #1
LABEL ST_MOVE	'Label to repeat move
FOL NEW_CYC TRIG1 * * *	'New master cycle (counter at 0) on trigger #1
FOL MOVEWT TRIG1 * * *	'Wait on next move until trigger 1 is active
FOL RATIO 1:1 * * *	'Following ratio is back to 1 to 1
MOVE SLEWCW * * *	'Start continuous following move on trigger #1
FOL MOVEWT 25 * * *	'Wait on next move for master position 25
FOL RATIO .5:1 * * *	'Set new following ratio
MOVE SLEWCW * * *	'Move to new following ratio at master position 25
FOL MOVEWT 75 * * *	'Wait on next move for master position 75
FOL RATIO 0:1 * * *	'Set new following ratio for slave to stop
MOVE SLEWCW * * *	'Move to new following ratio at master position 75
GOTO ST_MOVE	'Repeat the cycle
DONE	'End of program

FOL MDIST

Name	FOL MDIST
Descriptor	Master Move Distance
Type	Set-Up
Initial value	0
Range	±99999999 master steps
Default	FOL MDIST * * * *
Syntax	FOL MDIST 250.32 Q2 * 0.0
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

If a slave is doing preset moves, the FOL MDIST statement indicates the master distance over which the next preset moves will take place. Or, if a slave is in continuous mode, FOL MDIST is the master distance over

which acceleration or deceleration from the current ratio to the new ratio takes place. **FOL MDIST** is specified in user units and is scaled by the **UNIT MASTER** parameter. Numeric Q variables can be used with this statement.

In 4000 Following, acceleration for a slave can be specified either by master distance (**FOL MDIST**), or by the **ACCEL** statement. Whichever of these statements most recently precede the **MOVE** or **MOVI** statement will give the parameter used to determine the move profile. Specifying an acceleration for the slave means that the acceleration ramp is time-based and there is no position relationship between master and slave until the commanded following ratio is reached. Specifying a master distance for the slave's move profile ensures a precise position relationship between master and slave during all phases of the profile. Whenever the position relationship between master and slave is important, the **FOL MDIST** method should be used.

If a slave is in continuous mode and the master is starting from rest, setting **FOL MDIST** to 0 will ensure precise tracking of the master's acceleration ramp. The example below illustrates how this might take place.

See Also: **FOL ENABLE**, **UNIT MASTER**, **FOL RATIO**

Example

Statement	Description
UNIT MASTER 4000 * * *	'Master scale factor is 4000 steps/rev
FOL MASTER ENCL * * *	
FOL MDIST 0 * * *	'Assign following acceleration distance to 0 'master revs, i.e., instantaneous
FOL RATIO 1 * * *	'Set following ratio to 1 to 1
FOL ENABLE YES * * *	'Enable following on axis #1
MOVE SLEWCCW * * *	'Begin following master, if the master is 'not moving, slave will remain at rest until 'master moves. At this time it will track 'master precisely

FOL MOVEWT

Name	FOL MOVEWT					
Descriptor	Wait for Trigger or Cycle Position, Next Move					
Type	Programming					
Initial value	NO					
Range	TRIG 1-4, ±99999999 master steps					
Default	FOL MOVEWT * * * *					
Syntax	FOL MOVEWT TRIG1 NO 236.50 Q4					
Options	TAB	Q	NULL	TRIG	NO	
	F1	F2	F3	F4	F5	F6

Description

The **FOL MOVEWT** statement does not cause the 4000 to suspend program execution immediately. It specifies that the next move will not start until the specified trigger or master cycle position has been reached. This statement is useful in delaying subsequent moves until the master has reached the required position or an object has been sensed. When a master cycle position is specified, it is always an *absolute* position relative to the start of the last master cycle.

If a **MOVEWT** has been specified, and the 4000 program reaches a **MOVE** statement, the following sequence of events takes place. Moves for axes which do not have a wait condition specified will start and complete their moves as normal. This includes axes which are not configured as slaves. Axes that have a wait condition specified will wait for that condition to be satisfied before beginning their moves. The next program statement will execute when all preset moves (following or non-following) are complete and/or the cruise velocity has been reached for continuous moves.

For a **MOVI** statement after a **MOVEWT**, the program action taken is quite different. Axes that have no wait condition specified, or are not configured as slaves, will begin their moves. Axes that have a wait condition specified will wait to begin their moves until the condition becomes true, but program execution will continue immediately.

The Model 4000 records the **MOVEWT** condition as soon as the statement is executed, but will not start checking and waiting for the **MOVEWT** condition until the **MOVE** or **MOVI** statement is executed. If the condition is a trigger, and the trigger input goes active before the **MOVE** or **MOVI** statement is executed, it will not be noticed, and the move will wait for the next trigger. If **FOL MOVEWT NO** is given, or if a **MOVE STOP** or **MOVE KILL** is executed, all wait conditions for that axis are cleared, and the next **MOVE** or **MOVI** for that axis begins immediately. If the **MOVEWT** condition is a master cycle position, and that position has already been exceeded before the **MOVE** or **MOVI** statement is executed, that axis flags a **MOVEWT MISSED** condition. This condition can be detected with the **ON WT_ERR** statement, and is cleared by the next **FOL MOVEWT** command for that axis.

If a master cycle position is used in a **FOL MOVEWT** statement, the wait may occur while a master cycle is pending definition or a trigger. In that case the wait will include the trigger, then the cycle position in the master cycle defined by that trigger. If the pending status is cleared by either a **FOL NEWCYC NO** or a **FOL NEWCYC IMMED**, the wait is cleared also.

If the master cycle position specified with **FOL MOVEWT** is larger than the master cycle length, (**FOL MAS_CYC**), the 4000 simply waits for the specified position relative to the start of the current cycle. For example, if the master cycle length is 10 inches, current master cycle position is 5, and an **FOL MOVEWT** is issued for master cycle position 38 inches, the 4000 will wait for 33 more master inches before starting the next move. The value 38 is not referenced from where the master is currently, but rather total master inches from the start of the current cycle, even if it is necessary to wait for more than one complete cycle length. The actual master cycle position counter is still being reset to 0 when the master cycle is complete, but the **FOL WAIT** and **MOVEWT** will allow waiting for more than 1 cycle if necessary.

The master cycle position specified is in terms of user units and is scaled by the **UNIT MASTER** parameter. Numeric Q variables may be used to specify master cycle positions. Note that a trigger input number does not need to match the axis number to which the **FOL MOVEWT** is assigned. Also, keep in mind that an axis does not need to be in following mode (**FOL ENABLE YES**) to utilize the **FOL MOVEWT** and master cycle concept. However, a master must have been previously assigned using the **FOL MASTER** statement.

If cam profiling is enabled (**FOL CAM YES**), master cycle positions are determined by execution of the profile. For this reason, master cycle positions *may not* be used as a parameter with **FOL MOVEWT** when cam profiling is enabled. For a complete discussion of master cycle parameters with cam profiling, please refer to the section titled *Profiles and Master Cycles*.

The section titled *Following Wait Statements* provides general information about **FOL WAIT** and **FOL MOVEWT**, as well as the differences in their use. The *Continuous Cut to Length* example in that section uses **FOL MOVEWT 0** to ensure precise cut lengths.

See Also: **FOL MAS_CYC**, **FOL NEWCYC**, **FOL WAIT**, **FOL CYC_OFF**, **IN FOL TRIG**, **ON WT_ERR**

FOL NEWCYC

Name	FOL NEWCYC					
Descriptor	Define Master Cycle					
Type	Programming					
Initial value	NO					
Range	TRIG 1-4					
Default	FOL NEWCYC * * * *					
Syntax	FOL NEWCYC TRIG1 TRIG3 IMMED *					
Options	TAB	TRIG	NULL	IMMED	NO	
	F1	F2	F3	F4	F5	F6

Description

The **FOL NEWCYC** statement defines the beginning of a master cycle by setting the master cycle position to the value most recently specified with **FOL CYC_OFF**. If **IMMED** is specified, the master cycle position is set immediately. If a trigger is specified as the parameter, the 4000 will record the instruction to set the master cycle position when the specified trigger occurs. In either case, program flow continues normally. In the latter case, the master cycle is pending definition on the specified trigger, even though statements continue to execute. If an application requires suspending program flow until the trigger or a subsequent master cycle position occurs, the **FOL WAIT** statement may be used. **FOL NEWCYC NO** and **FOL NEWCYC IMMED** will remove the pending status of the master cycle definition. In this case, the former master cycle definition becomes effective again, and the specified trigger will not cause a new cycle definition. This also clears or undoes any **FOL WAIT** or **FOL MOVEWT** for master cycle position which was issued while the new cycle definition was pending a trigger input.

A new cycle automatically occurs, i.e., the master cycle position is set to 0, when the master cycle length (**FOL MAS_CYC**) is reached, even if no **FOL NEWCYC** statement has been executed.

If cam profiling is enabled (**FOL CAM YES**), the beginning of a master cycle is determined by execution of the profile. When cam profiling is enabled, The **FOL NEWCYC** statement is used to mark the repetitive portion of a cycle, and may only use the **IMMED** parameter. For a complete discussion of master cycle parameters with cam profiling, please refer to the section titled *Profiles and Master Cycles*.

See Also: **FOL MAS_CYC**, **FOL WAIT**, **FOL MOVEWT**, **FOL CYC_OFF**, IN **FOL TRIG**

FOL MAS_CYC

Name	FOL MAS_CYC					
Descriptor	Master Cycle Length					
Type	Set-Up					
Initial value	Ø					
Range	Ø - 99999999 master steps					
Default	FOL MAS_CYC * * * *					
Syntax	FOL MAS_CYC * 100 4.737 Q8					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL MAS_CYC** statement defines the length of the master cycle in user units. This value is scaled by the **UNIT MASTER** parameter. Numeric **Q** variables can be used with this statement. The initial value for **FOL MAS_CYC** is 0, which means that the default master cycle length is infinite.

The concept of a master cycle may be useful when moves or other events must be initiated at certain master positions. By specifying a master cycle length, periodic actions may be programmed in a loop or with subroutines which refer to cycle positions, even if the master runs continuously. It is possible to program the 4000 to suspend program operation or wait for moves until certain master cycle positions or trigger inputs. The master cycle length, **FOL MAS_CYC**, should be defined before the functions which wait for periodic master cycle positions are used. An axis need not be in following

mode (i.e., `FOL ENABLE YES`) to utilize the concept of a master cycle. However, a master must have been previously assigned with the `FOL MASTER` statement.

If cam profiling is enabled (`FOL CAM YES`), the master cycle length is determined by the sum of the master distances in the repetitive portion of the profile. For this reason, the `FOL MAS_CYC` statement is ignored when cam profiling is enabled. For a complete discussion of master cycle parameters with cam profiling, please refer to the section titled *Profiles and Master Cycles*.

See Also: `FOL NEWCYC`, `FOL WAIT`, `FOL MOVEWT`, `FOL CYC_OFF`

FOL WAIT

Name	FOL WAIT					
Descriptor	Wait for Trigger or Master Cycle Position					
Type	Programming					
Initial value	NO					
Range	TRIG 1-4, ±99999999 master steps					
Default	FOL WAIT * * * *					
Syntax	FOL WAIT TRIG2 NO 14.53 Q8					
Options	TAB	Q	NULL	TRIG	NO	
	F1	F2	F3	F4	F5	F6

Description

The `FOL WAIT` statement causes the 4000 to suspend program execution until the specified trigger or master cycle position has occurred. This function is useful in delaying subsequent I/O operation until the master has achieved the required position or an object has been sensed. When a master cycle position is specified, it is always an *absolute* position relative to the start of the last master cycle.

If the `FOL WAIT` condition is a master cycle position, and that position has already been exceeded when the `FOL WAIT` statement is executed, then program flow proceeds immediately to the next statement. No error condition results from this.

If a master cycle position is used in a `FOL WAIT` statement, the wait may occur while a master cycle is pending definition or a trigger. In that case the wait will include the trigger, then the cycle position in the master cycle defined by that trigger. If the pending status is cleared by either a `FOL NEWCYC NO` or a `FOL NEWCYC IMMED`, the wait is cleared also.

If the master cycle position specified with `FOL WAIT` is larger than the master cycle length, (`FOL MAS_CYC`) the 4000 simply waits for the specified position relative to the start of the current cycle. For example, if the master cycle length is 25 inches, current master cycle position is 5, and an `FOL WAIT` is issued for master cycle position 67, the 4000 will wait for 62 more inches of master travel before continuing program execution. The value 67 is not referenced from where the master is currently, but rather total master inches from the start of the current cycle, even if it is necessary to wait for more than one complete cycle length. Here, our wait was for about 2.5 master cycles. The actual master cycle position counter is still being reset to 0 when every master cycle is complete, but the `FOL WAIT` and `MOVEWT` will allow waiting for more than 1 cycle if necessary.

The master cycle position specified is in terms of user units and is scaled by the `UNIT MASTER` parameter. Numeric Q variables may be used to specify master cycle positions. Note that a trigger input number need not match the axis number to which the `FOL WAIT` is assigned. For example, axis #2 could be performing a `FOL WAIT`, waiting for Trigger 4. Also, keep in mind that an axis does not need to be in following mode (i.e., `FOL ENABLE YES`) to utilize the `FOL WAIT` and master cycle concept. However, a master must have been previously assigned using the `FOL MASTER` statement.

`FOL WAIT` is similar to other `WAITS` in that an `ON` condition may clear the `FOL WAIT` and allow subsequent statements to execute. A `FOL WAIT NO`

will also clear the wait. If one program is waiting on a FOL WAIT, another program may clear the wait by executing a FOL WAIT NO, assuming both programs are running under Multi Tasking

See Also: FOL MAS_CYC, FOL NEWCYC, FOL MOVEWT, FOL CYC_OFF, IN FOL TRIG

Example

In the example below, the master is an encoder mounted to gearing on a conveyor line. The gearing results in 16,000 encoder steps per conveyor inch. The slave on axis one is a 25,000 step/rev microstepper on a 36" long, 4 pitch leadscrew. The slave waits for the product to be sensed on the conveyor, accelerates to a 1 to 1 ratio, waits for a safe location to actuate the stamping equipment, then applies an inked stamp to the product at the correct location. After the stamp is placed, the slave quickly moves back to the starting position and waits for the next product. The example illustrates how the FOL WAIT command can be used to wait for master cycle positions in order to coordinate motion.

Statement	Description
UNIT VEL 100000 * * *	'Set slave velocity scale factor to 100000
UNIT ACCEL 100000 * * *	'Set slave accel scale factor to 100000
UNIT POS 100000 * * *	'Set slave position scale factor to 100000 to program in inches
ACCEL 10 * * *	'Acceleration = 10 inches/sec/sec
VEL 5 * * *	'Velocity = 5 inches/sec (non-following moves)
MODE M_ABS * * *	
UNIT MASTER 16000 * * *	'Set master scale factor to program in inches
FOL MASTER ENC1 * * *	'Assign encoder input #1 as master
FOL RATIO 1 * * *	'Following ratio 1 to 1 slave to master inch.
FOL MDIST 1 * * *	'Accelerate the slave over 1 master inch for following moves
FOL MAS_CYC 40 * * *	'Master cycle length is 40 inches
LABEL INK_ON	'Label to repeat inking process
FOL ENABLE YES * * *	'Enable following on axis #1
FOL NEWCYC TRIG2 * * *	'Begin new master cycle on trigger #2 (product sensed on conveyor)
FOL MOVEWT TRIG2 * * *	'Start next move when trigger #2 is active
MOVE SLEWCW * * *	'Start continuous slave move on trigger #2
FOL WAIT 10.5 * * *	'Wait until master position is 10.5 inches, this is when the stamping device can be actuated without mechanical damage to the leadscrew assembly
OUT BIT7 = 1	'Turn on actuator to place ink stamp on product
FOL WAIT 12.0 * * *	'Wait until master position is 12 inches. The ink stamp is pressed in place by a stationary roller 1.5" in length
OUT BIT7 = 0	'Turn actuator off
MOVE STOP * * *	'Stop slave move
FOL ENABLE NO * * *	'Disable following on axis #1
MOVE 0 * * *	'Move back to home position
GOTO INK_ON	'Begin cycle again on trigger #2
DONE	'End of program

FOL SMOOTH

Name	FOL SMOOTH					
Descriptor	Velocity Measurement Smoothing					
Type	Set-Up					
Initial value	1					
Range	1-4					
Default	FOL SMOOTH * * * *					
Syntax	FOL SMOOTH 4 2 * 1					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL SMOOTH** statement specifies the degree to which the Model 4000 filters and smoothes the measurements of master position and velocity. Valid choices of 1, 2, 3, and 4 for the **FOL SMOOTH** statement correspond to velocity averaging periods of 4, 8, 16, and 32 milliseconds, respectively.

The 4000 samples the master position every 2 msec, and by measuring the change in master position over a fixed number of samples, the master velocity is calculated. The current master position, the current ratio, and the estimation of current master velocity are all used to calculate the setpoint slave position for the next sample. If left unspecified, the following algorithm defaults to **FOL SMOOTH 1**.

For applications in which the master is vibrating or moving very slowly, smoother slave velocity may result with an **FOL SMOOTH** parameter greater than 1. When the master is changing velocity very quickly, or if the number of master pulses per sample period is large, an **FOL SMOOTH** value of 1 is recommended. Setting this value will best be done on a trial and error basis, since each application has widely varying components and requirements.

See Also: **FOL MAXVEL**, **FOL MAXACC**, **FOL VELFF**

FOL MAXACC

Name	FOL MAXACC					
Descriptor	Slave Maximum Acceleration					
Type	Set-Up					
Initial value	Maximum for motor resolution					
Default	FOL MAXACC * * * *					
Syntax	FOL MAXACC 200 Q3 * *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL MAXACC** statement sets the maximum acceleration for slave axes. The **FOL MAXACC** statement accepts numeric Q variables as an argument and is scaled by the **UNIT ACCEL** parameter.

The profile of the acceleration for a slave axis can be defined with one of two statements: **ACCEL** or **FOL MDIST**. If the **ACCEL** command is used to determine slave acceleration, the acceleration ramp will be a time-based acceleration at the value specified, as long as the **ACCEL** value is not larger than **FOL MAXACC**. If the **FOL MDIST** command is used to define a move profile, the resulting acceleration will be determined in part by the speed of the master, however, the acceleration ramp the slave takes will never exceed the **FOL MAXACC** value.

For both cases above, if the required acceleration is larger than **FOL MAXACC**, the slave will begin falling behind its commanded position. The 4000 will attempt to make up this position error as soon as the commanded accel falls below **FOL MAXACC**. An error correction velocity is added to that implied by the ratio setpoint. The velocity used to make up the error is limited to that specified with **POSM MAXVEL**.

As with **FOL MAXVEL**, **FOL MAXACC** should be determined and defined early in the development stage of an application to prevent any damage to the load on the slave axis when unexpectedly high accelerations are

commanded. The torque available from the slave motor will also be a determining factor in this parameter in order to prevent motor stalls.

See Also: FOL SMOOTH, FOL MAXVEL, FOL VELFF

FOL MAXVEL

Name	FOL MAXVEL					
Descriptor	Slave Maximum Velocity					
Type	Set-Up					
Initial value	Maximum for motor resolution					
Range	Limited by slave motor resolution maximum velocity					
Default	FOL MAXVEL * * * *					
Syntax	FOL MAXVEL 100 Q2 * *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The FOL MAXVEL statement sets the maximum velocity at which slave axes may travel. The FOL MAXVEL statement accepts numeric Q variables as an argument and is scaled by the UNIT VEL parameter.

Normally in a following application, the slave velocities will be known based on the normal speed of the master and the commanded following ratios. In some cases, however, the master speed may be higher than normal, the slave may be commanded to perform a shift move, or some other event may occur which will cause the slave to travel at a velocity higher than expected. In these cases, the 4000 will raise the speed of the slave as necessary to perform the required move, but only up to the FOL MAXVEL. If the commanded speed is higher than FOL MAXVEL, the slave axis will start falling behind its commanded position. The 4000 will attempt to make up this position error as soon as the commanded speed falls below FOL MAXVEL. An error correction velocity is automatically added to that implied by the ratio setpoint. The velocity used to make up the error is limited to that specified with POSM MAXVEL.

The FOL MAXVEL should be determined and defined early in the development stage of an application to prevent any damage to the load on the slave axis when unexpectedly high velocities are commanded.

See Also: FOL SMOOTH, FOL MAXACC, FOL VELFF

FOL VELFF

Name	FOL VELFF					
Descriptor	Enable or Disable Velocity Feed Forward					
Type	Set-Up					
Initial value	YES					
Default	FOL VELFF YES YES YES YES					
Syntax	FOL VELFF YES NO * *					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The FOL VELFF statement allows the user to enable or disable velocity feed forward in the 4000 Following algorithm. Velocity feed forward is activated by default in the Following algorithm, but can be turned off as desired with the FOL VELFF statement.

The 4000 measures master position every two milliseconds, and calculates a corresponding slave setpoint. This calculation and achieving the subsequent slave setpoint position require 4 milliseconds. Enabling velocity feed-forward (FOL VELFF YES) eliminates any lag in slave position which would be dependent on master speed. It may be desirable to deactivate velocity feed forward when maximum slave smoothness is important and minor phase delays can be accommodated. A detailed discussion of velocity feed-forward is given in the section *Technical Considerations for Following*.

See Also: FOL SMOOTH, FOL MAXACC, FOL MAXVEL

FOL CYC_OFF

Name	FOL CYC_OFF					
Descriptor	Master Cycle Offset Position					
Type	Set-Up					
Initial value	0					
Range	±99999999 motor steps					
Default	FOL CYC_OFF * * * *					
Syntax	FOL CYC_OFF * -10.2 Q8 *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL CYC_OFF** statement defines the initial master cycle position in user units. The initial master cycle position is assigned with a new master cycle defined via the **FOL NEWCYC** statement. This value is scaled by the **UNIT MASTER** parameter. Numeric **Q** variables can be used with this statement. The default value for **FOL CYC_OFF** is 0, which means that the master cycle position will be 0 upon definition of a new master cycle (see **FOL NEWCYC**).

The concept of an initial master cycle offset may be useful if new master cycle definition must take place at a master position which is different from what needs to be considered the beginning of a periodic cycle. The initial position defined with **FOL CYC_OFF** applies to the first cycle only. When a master cycle is complete, the master cycle position rolls over to zero. A negative offset would be used if some master travel were desired before master cycle position was zero. A positive offset would be used if it was necessary to enter the first master cycle at a position other than 0. For example, suppose **FOL MAS_CYC** was set to 20 and **FOL CYC_OFF** was set to 7. When the **FOL NEWCYC** definition takes place, either via **FOL NEWCYC IMMED** or the specified trigger, the initial master cycle position will be 7. Rollover will occur after the master travels 13 more units, and the master cycle position would go to zero.

If cam profiling is enabled (**FOL CAM YES**), the master cycle offset position is determined by the sum of the master distances in the lead in portion of the profile. For this reason, the **FOL CYC_OFF** statement is ignored when cam profiling is enabled. For a complete discussion of master cycle parameters with cam profiling, please refer to the section titled *Profiles and Master Cycles*.

See Also: **FOL NEWCYC**, **FOL MAS_CYC**, **FOL_WAIT**, **FOL_MOVEWT**

FOL M_SYNC

Name	FOL M_SYNC					
Descriptor	Define Master Synchronization Mark					
Type	Set-Up					
Initial value	NO					
Range	Trig 1-4, 0 - 99999999 master steps					
Default	FOL M_SYNC * * * *					
Syntax	FOL M_SYNC TRIG1 Q1 * 50					
Options	TAB	Q	NULL	TRIG	NO	
	F1	F2	F3	F4	F5	F6

Description

The **FOL M_SYNC** statement defines an event which will cause the slave position to be read and saved for future periodic synchronization calculations. This external event may be any of the four trigger inputs TRIG1 through TRIG4, or a master cycle position. If a master cycle position is used, the value is scaled by the **UNIT MASTER** parameter. Numeric **Q** variables may be used with this statement. The default value for **FOL M_SYNC** is that the master sync mark is not defined, which means that the periodic synchronization features may not be used.

There are many applications in which periodic operations must occur in intervals which are not perfectly repeatable. The 4000 allows the user to define two external events, or "marks", which capture the slave position.

These are called *Master Sync Mark* and *Slave Sync Mark*, and are defined with the `FOL M_SYNC` and `FOL S_SYNC` statements respectively. If the sync mark is defined as a trigger, the slave position is captured on each occurrence of that trigger. If the sync mark is defined a master cycle position, the slave position is captured *on only the first occurrence of that cycle position* each time a new cycle is defined, or a master cycle has completed and rolled over to start a new cycle. For a complete understanding of a sync mark definitions and their use in periodic synchronization, please refer to the section titled *Periodic Master/Slave Synchronization* earlier in this chapter.

See Also: `FOL M_SYNC`, `FOL SYNC_OFF`, `IN Qn = FOL AXISn SYN_ERR`, `IN Qn = FOL AXISn M_SYNC`

FOL S_SYNC

Name	FOL S_SYNC					
Descriptor	Define Slave Synchronization Mark					
Type	Set-Up					
Initial value	NO					
Range	Trig 1-4, ±99999999 slave steps					
Default	FOL S_SYNC * * * *					
Syntax	FOL S_SYNC TRIG1 Q1 * 50					
Options	TAB	TRIG	Q	NO	NULL	
	F1	F2	F3	F4	F5	F6

Description

The `FOL S_SYNC` statement defines an event which will cause the slave position to be read and saved for future periodic synchronization calculations. This external event may be any of the four trigger inputs TRIG1 through TRIG4, or a master cycle position. If a master cycle position is used, the value is scaled by the `UNIT MASTER` parameter. Numeric Q variables may be used with this statement. The default value for `FOL S_SYNC` is that the slave sync mark is not defined, which means that the periodic synchronization features may not be used.

There are many applications in which periodic operations must occur in intervals which are not perfectly repeatable. The 4000 allows the user to define two external events, or marks, which capture the slave position. These are called *Master Sync Mark* and *Slave Sync Mark*, and are defined with the `FOL M_SYNC` and `FOL S_SYNC` statements respectively. If the sync mark is defined as a trigger, the slave position is captured on each occurrence of that trigger. If the sync mark is defined as a master cycle position, the slave position is captured *on only the first occurrence of that cycle position* each time a new cycle is defined, or a master cycle has completed and rolled over to start a new cycle. For a complete understanding of a sync mark definitions and their use in periodic synchronization, please refer to the section titled *Periodic Master/Slave Synchronization* earlier in this chapter.

See Also: `FOL M_SYNC`, `FOL SYNC_OFF`, `IN Qn = FOL AXISn SYN_ERR`, `IN Qn = FOL AXISn S_SYNC`

FOL SYNC_OFF

Name	FOL SYNC_OFF					
Descriptor	Define Expected Synchronization Position Difference					
Type	Set-Up					
Initial value	0					
Range	±99999999 slave steps					
Default	FOL SYNC_OFF * * * *					
Syntax	FOL SYNC_OFF Q2 10 -5 *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL SYNC_OFF** statement defines the expected synchronization offset in the user's slave position units. This value is scaled by the **UNIT POS** parameter. Numeric **Q** variables may be used with this statement. The default value for **FOL SYNC_OFF** is 0, which means that a master and slave sync marks are expected to occur simultaneously.

Each time either a master or slave sync mark occurs, the corresponding slave position is captured and saved internally. The **FOL SYNC_OFF** statement defines the expected difference between these captured slave positions. This expected difference is called the *Slave Synchronization Offset*. By defining the offset expected between two positions instead of defining the position expected at a single synchronization mark, continuous motion in one direction is allowed without requiring a continuous re-calculation of the expected slave position. The difference between the actual offset and the expected offset is called the Sync Error. This error may be read into a **Q** variable using the **IN Qn = FOL AXISn SYN_ERR** statement. To understand exactly how to use this, please refer to the section titled *Periodic Master/Slave Synchronization* earlier in this chapter.

See Also: **FOL M_SYNC**, **FOL S_SYNC**, **IN Qn = FOL AXISn SYN_ERR**

FOL WIN_P

Name	FOL WIN_P					
Descriptor	Define Master Window Position					
Type	Set-Up					
Initial value	0					
Range	0 - 99999999 master steps					
Default	FOL WIN_P * * * *					
Syntax	FOL WIN_P Q1 10 * *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOL WIN_P** statement defines the position of a following error detection window in the user's master position units. This value is scaled by the **UNIT MASTER** parameter. Numeric **Q** variables may be used with this statement. The default value for **FOL WIN_P** is 0, which means that the start of a following error detection window would coincide with the start of a master cycle.

The concept of an error detection window is useful in applications in which a periodic repetitive operation takes place, and precise synchronization is only important during a portion of the cycle. For such an application, the window defines the portion of a cycle in which excess following error is detected, while being ignored in the remainder of the cycle. For a complete discussion of the conditions which may result in following error, please refer to the section titled *Following Performance* earlier in this chapter. The error detection window start position must be less the master cycle length to be valid, i.e., meaningful. If its value is greater than master cycle length, the error detection window will not be valid, and *error detection will occur continuously*. Programmed response to detection of following error is

enabled through use of the `ON FOL_ERR` statement. The `FOL_WIN_P` statement may be issued at any time, even while motion is in progress, and takes effect immediately. For a complete discussion of error detection windows and related topics, please refer to the section titled *Monitoring Following Error* earlier in this chapter.

See Also: `FOL_PTOL`, `FOL_WIN_W`, `ON FOL_ERR`, `IN Qn = FOL_AXISn FOL_ERR`

FOL_WIN_W

Name	FOL_WIN_W					
Descriptor	Define Master Window Width					
Type	Set-Up					
Initial value	Ø					
Range	0 - 99999999 master steps					
Default	FOL_WIN_W * * * *					
Syntax	FOL_WIN_W Q3 2 * *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The `FOL_WIN_W` statement defines the width of a following error detection window in the user's master position units. This value is scaled by the `UNIT_MASTER` parameter. Numeric `Q` variables may be used with this statement. The default value for `FOL_WIN_W` is 0, which means that the following error detection window is not valid.

The concept of an error detection window is useful in applications in which a periodic repetitive operation takes place, and precise synchronization is only important during a portion of the cycle. For such an application, the window defines the portion of a cycle in which excess following error is detected, while being ignored in the remainder of the cycle. For a complete discussion of the conditions which may result in following error, please refer to the section titled *Following Performance* earlier in this chapter. The error detection window width must be less the master cycle length and greater than zero to be valid, i.e., meaningful. If its value is less than master cycle length or zero, the error detection window will not be valid, and *error detection will occur continuously*. Programmed response to detection of following error is enabled through use of the `ON FOL_ERR` statement. The `FOL_WIN_W` statement may be issued at any time, even while motion is in progress, and takes effect immediately. For a complete discussion of error detection windows and related topics, please refer to the section titled *Monitoring Following Error* earlier in this chapter.

See Also: `FOL_PTOL`, `FOL_WIN_P`, `ON FOL_ERR`, `IN Qn = FOL_AXISn FOL_ERR`

FOL_PTOL

Name	FOL_PTOL					
Descriptor	Define Following Error Tolerance					
Type	Set-Up					
Initial value	Ø					
Range	0 - 99999999 slave steps					
Default	FOL_PTOL * * * *					
Syntax	FOL_PTOL Q4 * .01 *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The `FOL_PTOL` statement defines the following position error tolerance in the user's slave position units. This value is scaled by the `UNIT_POS` parameter. Numeric `Q` variables may be used with this statement. The default value for `FOL_PTOL` is 0, which means that a following position error of any magnitude may be detected.

The concept of an allowable position following error is useful when there is a limit to the acceptable following error in the course of following operation.

Mechanical constraints as well as those imposed by user programming may cause the actual position of the slave to differ from the commanded position. For a complete discussion of the conditions which may result in following error, please refer to the section titled *Following Performance* earlier in this chapter. Programmed response to the detection of following error is enabled through use of the ON FOL_ERR statement, and may be limited to specific portions of a master cycle. If the error value ever exceeds the tolerance given with FOL_PTOL when error detection is enabled, the 4000 will branch to the location specified in the ON FOL_ERR statement. The FOL_PTOL statement may be issued at any time, even while motion is in progress. The new value takes effect immediately, and also clears any previously detected error. This avoids unwanted detection of previous errors. For a complete discussion of related topics, please refer to the section titled *Monitoring Following Error* earlier in this chapter.

See Also: FOL_WIN_P, FOL_WIN_W, ON FOL_ERR, IN Qn = FOL_AXISn FOL_ERR

FOL LEAD

Name	FOL LEAD					
Descriptor	Define Setpoint Lead Time					
Type	Set-Up					
Initial value	Ø milliseconds					
Range	0-250 milliseconds					
Default	FOL LEAD * * * *					
Syntax	FOL LEAD 0.82 * Q1 *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The FOL_LEAD statement defines the lead time for the specified axes. Some drives, such as the Dynaserv step and direction versions, have up to 6 milliseconds of lag, i.e., delayed response to step and direction input. This creates a velocity dependent position lag, which causes a synchronization error. In following, however, every position along the profile is important, because it must correspond to a master position. The setpoint lead feature allows the 4000 to dynamically advance the slave setpoint beyond what would be normally resulting from the profile. It is advanced by the product of the instantaneous velocity and the lead value specified with the FOL_LEAD statement. Because of possible instability and profile skew, it is best to avoid this feature when possible, i.e., use the default value of zero. If it is necessary to compensate for drive lag, the proper value will probably need to be determined empirically.

See Also: FOL_DIRSET, FOL_ENCCHK

FOL DIRSET

Name	FOL DIRSET					
Descriptor	Enable direction change setup time					
Type	Setup					
Initial value	YES					
Range	FOL DIRSET * * * *					
Default	FOL DIRSET YES NO * YES					
Syntax	FOL LEAD 0.82 * Q1 *					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The FOL_DIRSET statement enables or disables the default 2 millisecond direction input change setup time of the 4000. Most Compumotor steppers require some minimum time for the drive to respond to a change on the direction input before any steps are given in the new direction. In normal positioning applications, motion always stops before a move is commanded in the opposite direction, so this requirement is of no consequence. In following however, the master may change direction, resulting in a direction

change on the slave without a new move command. When the setup time is enabled, the 4000 temporarily saves the steps which would have been sent with the direction change. These steps are sent during the next 2 millisecond update. Some drives, such as the Dynaserv and Z drives, do not have a direction change setup requirement. In order to facilitate smooth following on these drives, the **FOL DIRSET** statement may be used to disable the default 2 millisecond direction change setup.

See Also: **FOL LEAD**, **FOL ENCCHK**

FOL ENCCHK

Name	FOL ENCCHK					
Descriptor	Enable encoder and motor step comparison					
Type	Setup					
Initial value	NO					
Range	FOL ENCCHK * * * *					
Default	FOL ENCCHK YES NO * YES					
Syntax	FOL LEAD 0.82 * Q1 *					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The **FOL ENCCHK** statement enables or disables the comparison between the commanded position in motor steps, i.e. controlled in motor step mode, and the actual position in encoder steps. This is especially useful when the slave is a Compumotor servo drive such as the Z drive or the Dynaserv. Its first major purpose is to assist in the tuning of the drive for minimum following error. It also facilitates rapid electronic response to excess following error. For a complete discussion of the purposes and uses of this feature, please refer to the section titled *Motor and Encoder Comparison*.

See Also: **FOL LEAD**, **FOL DIRSET**

FOL CAM

Name	FOL CAM					
Descriptor	Enable cam profiling					
Type	Programming					
Initial value	NO					
Range	FOL CAM * * * *					
Default	FOL CAM YES NO * YES					
Syntax	FOL LEAD 0.82 * Q1 *					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The **FOL CAM** statement enables or disables the cam profiling mode of following motion. Cam profiling is used to delete and create pre-defined motion profiles. These profiles may then be executed at very high cycle rates. When cam profiling is enabled, motion definition and initiation statements have changed meaning. Master cycle parameters also change. For a complete discussion of the purposes and uses of this feature, please refer to the section titled *Cam Profiling*.

See Also: **FOL MAS_CYC**, **FOL CYC_OFF**, **FOL NEWCYC**

FOLM

Name	FOLM					
Descriptor	Moving Positioning System Parameters					
Type	Set-Up					
Default	N/A					
Syntax	FOLM					
Options	TAB	DEF	RATIO	PDEF	ENABLE	
	F1	F2	F3	F4	F5	F6

Description

FOLM statements define the moving positioning system (MPS). The MPS allows point-to-point, contouring, ratio following, and many other types of moves to be super-imposed on the slave following the master. All of these moves are programmed as if the slave was standing still, making implementation of the MPS very easy. Below is a summary of the moving positioning system statements.

- FOLM PDEF** Define slave's initial position when MPS is defined.
- FOLM RATIO** Establish the ratio of slave to master required to keep the slave stationary with respect to the master.
- FOLM DEF** Define the MPS. This involves setting the master cycle count and position to 0 and the slave position to that specified with the FOLM PDEF statement.
- FOLM ENABLE** Enable the moving positioning system. Subsequent slave moves are now with respect to the MPS coordinate system.

See Also: FOL, UNIT MASTER

FOLM PDEF

Name	FOLM PDEF					
Descriptor	Moving Positioning Offset					
Type	Set-Up					
Initial value	0					
Range	±99999999 slave steps					
Default	FOLM PDEF * * * *					
Syntax	FOLM PDEF 0 Q7 12.35 *					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **FOLM PDEF** statement is used to define the slave's initial position when the moving positioning system is defined. This parameter is entered in user units, and is scaled by the **UNIT POS** parameter. This number will be used each time the MPS is defined, so it must be defined before the MPS is defined for the first time. Numeric Q variables can be entered for this statement, and the default value for **FOLM PDEF** is 0.

The Moving Positioning System in the 4000 allows programming of slave moves with respect to moving reference. Point-to-point, contouring, even ratio following moves can be performed by the slave while it's following the master. For more information on the moving positioning system, refer to the example under the **FOLM ENABLE** statement.

See Also: FOLM RATIO, FOLM DEF, FOLM ENABLE, UNIT POS

FOLM RATIO

Name	FOLM RATIO					
Descriptor	Moving Positioning Ratio					
Type	Set-Up					
Initial value	0					
Range	±127 slave steps per master step					
Default	FOLM RATIO * * * *					
Syntax	FOLM RATIO 1:1 -.5:1 Q2:Q9 *					
Options	TAB	Q	NULL	:		
	F1	F2	F3	F4	F5	F6

Description

The **FOLM RATIO** statement establishes the ratio of slave to master required to keep the slave *stationary* in the moving positioning system. Assume **FOLM RATIO** is set to .5:.3 for an axis. The first parameter is

scaled by the `UNIT POS` value to give slave steps. The second parameter is scaled by the `UNIT MASTER` value to give master steps. For a `UNIT POS` of 25000 and a `UNIT MASTER` of 4000, the slave to master step ratio would be $.5 * 25000$ to $.3 * 4000$, or 125 slave steps for every 12 master steps.

If no second parameter is specified, it is assumed to be 1. Numeric Q variables can be used with this statement for slave and/or master parameters. The first, or slave, parameter is a signed number, which will be negative if the slave counts in the opposite direction from the master. This statement must be executed before the moving positioning system is defined by the `FOLM DEF` statement. For more information on the moving positioning system, refer to the *Moving Positioning System* section earlier in this chapter, and the example under the `FOLM ENABLE` statement.

See Also: `FOLM PDEF`, `FOLM DEF`, `FOLM ENABLE`, `UNIT MASTER`, `UNIT POS`

FOLM DEF

Name	FOLM DEF					
Descriptor	Define Moving Positioning System					
Type	Programming					
Initial value	NO					
Range	TRIG 1-4					
Default	FOLM DEF * * * *					
Syntax	FOLM DEF TRIG1 TRIG3 IMMED *					
Options	TAB	TRIG	NULL	IMMED	NO	
	F1	F2	F3	F4	F5	F6

Description

The `FOLM DEF` statement defines the moving positioning system (MPS). When the MPS is defined the 4000 establishes the position reference for master and slave with respect to the MPS. At the time of MPS definition, the slave position within the MPS is set to that defined with the `FOLM PDEF` statement. If the application requires that the master cycle position also be set to zero at the time of the MPS definition, the `FOL NEWCYC` statement should be used. The same trigger could be used to initiate both MPS definition and a new master cycle.

If `IMMED` is specified, the MPS is defined immediately. If a trigger is specified, the 4000 defines the moving positioning system when the trigger occurs. Program flow is not affected in either case. If the application requires that the program be halted until a specific master cycle position or the trigger occurs, the `FOL WAIT` and `FOL MOVEWT` statements may be used.

Just because the MPS is defined, it does not mean that subsequent slave moves will be with reference to the MPS. The `FOLM ENABLE` statement allows reference to the stationary or moving positioning systems. For more information on the moving positioning system, refer to the *Moving Positioning System* section earlier in this chapter, and the example under the `FOLM ENABLE` statement.

See Also: `FOLM PDEF`, `FOLM RATIO`, `FOLM ENABLE`, `IN FOL TRIG`

FOLM ENABLE

Name	FOLM ENABLE					
Descriptor	Enter or Exit Moving Positioning System					
Type	Programming					
Initial value	NO					
Default	FOLM ENABLE * * * *					
Syntax	FOLM ENABLE YES NO * YES					
Options	TAB	YES	NULL	NO		
	F1	F2	F3	F4	F5	F6

Description

The **FOLM ENABLE** statement is used to enter and exit the moving positioning system (MPS). The MPS must have been previously defined when this command is executed, and executing this command does not affect the master and slave position coordinates already established at the time the **FOLM DEF** command was executed. Even if the moving positioning system is not entered right after definition, the master and slave positions are constantly kept track of in **both** positioning systems. This ensures that no position information is lost when transferring from the stationary positioning system to the moving positioning system and back.

Entering and exiting the MPS simply means taking the point of view of the moving object or stationary reference, respectively. It does not affect the shaft speed of the slave. If an application requires that the slave start tracking the master immediately upon entry into the MPS (i.e., remain stationary in the MPS), a **MOVE STOP** or **MOVE** to a position should be commanded right after the MPS is defined and entered.

If the slave is in absolute mode, move commands issued while in the MPS will be performed with respect to the moving absolute position, and those executed while not in the MPS will be performed with respect to the stationary absolute position. The 4000 keeps track of both absolute coordinate systems. The moving positioning system can be illustrated more clearly with the example below.

See Also: **FOLM PDEF**, **FOLM RATIO**, **FOLM DEF**

Example

In the example below, a pattern of coordinates needs to be traced on a grid tray which is located on a moving conveyor belt. The part is moved with an XY stage, where the X axis will be following the speed of the conveyor. The third axis of the 4000 is controlling the conveyor, so the X axis will be following the step output of axis #3. Assume that the third motor is mounted to a pulley of 3" diameter driving the conveyor. The X and Y motors, 4000 axes 1 and 2, respectively, are coupled to 4 pitch leadscrews and are driven with drives of 25,000 step/rev resolution.

The XY stage will initially be homed and the absolute position set to 0. The operator enters the required pattern number, and a subroutine is called from another program location (not shown here) which determines the required values for Q variables used in the **MOVE** commands. The conveyor is set in motion and the trays start moving down the line.

When a tray is sensed, the moving positioning system is defined and entered immediately. When the system is defined, the X position is defined to be at 23.4". The trays are 20" in length and the home position of the X axis is 3.4" from the sensor which detects the leading edge of the tray. Setting our initial X coordinate to 23.4" means that the far edge of the tray is at position 0. This assumption was used in the subroutine **SET_VARS** where the coordinate location variables are set. After the MPS is entered by the X axis, the XY stage makes three linearly interpolated moves to the proper tray locations, turning on an actuator at each location. The X axis then exits the MPS and the XY stage returns to the home position to await the next tray.

<u>Statement</u>	<u>Description</u>
UNIT POS 100000 100000 25000 *	'X and Y positioning scale 'factors for inches, third axis 'motor resolution
UNIT VEL 100000 100000 1326.291 *	'Velocity scale factors for 'inches/sec
UNIT ACCEL 100000 100000 1326.291 *	'Acceleration scale factors for 'inches/sec/sec
VEL 5 5 12.25 *	'Velocity
ACCEL 10 10 35 *	'Acceleration
UNIT PATH VEL 100000	'Unit path velocity for LINT 'mode move
UNIT PATH ACCEL 100000	'Unit path acceleration for 'LINT mode move
PATH VEL 5	'Path velocity
PATH ACCEL 10	'Path acceleration
LINT MODE YES YES * *	'Enable LINT mode on axes #1 'and 2
UNIT MASTER 1326.2912 * * *	'Master scale factor for inches
FOL MASTER MOT3 * * *	'Master for axis #1 is step 'output of axis #3
FOLM RATIO 1:1 * * *	'MPS slave to master ratio is 1 'to 1
FOLM PDEF 23.4 * * *	'Define initial slave position 'to 23.4"
FOLM MAS_CYC 35 * * *	'Master cycle length is 35 'inches (well larger than the 'slave offset + tray length)
MOVE HOMECW HOMECCW * *	'Move XY stage to home
MODE M_ABS M_ABS * *	'Absolute mode
PDEF 0 0 * *	'Define home position as 0
IN Q1 = LCD1,1 ^ENTER PATTERN ^	'Operator enters required grid 'pattern
GOSUB SET_VARS	'Subroutine that contains the 'grid coordinates (i.e., values 'of Q2 - Q7) for the possible 'patterns
MOVE * * SLEWCW *	'Start conveyor moving
LABEL PLACE_PT	'Label to repeat part placement 'on tray
FOL NEWCYC TRIG3 * * *	'Set master position to 0 on 'trigger #3
FOLM DEF TRIG3 * * *	'Define MPS on trigger #3
FOL WAIT TRIG3 * * *	'Suspend program operation 'until 1st tray is sensed
FOLM ENABLE YES * * *	'Enter MPS on axis #1, does not 'start motor
MOVE STOP * * *	'Must stop the X axis within 'the MPS, since when the MPS 'was entered, we are at rest in 'the stationary reference 'frame, but actually in motion 'in the MPS reference frame - a 'LINT mode move can not be 'started while an axis is in 'motion.
MOVE Q2 Q3 * *	'Do LINT mode move to 1st grid 'coordinates
OUT POB3 = 1	'Actuate part placement device
WAIT FOR .2 SECONDS	'Wait for placement
OUT POB3 = 0	'Turn actuator back off
MOVE Q4 Q5 * *	'Do LINT mode move to 2nd grid 'coordinates
OUT POB3 = 1	'Actuate part placement device
WAIT FOR .2 SECONDS	'Wait for placement
OUT POB3 = 0	'Turn actuator back off
MOVE Q6 Q7 * *	'Do LINT mode move to last grid 'coordinates
OUT POB3 = 1	'Actuate part placement device

```

WAIT FOR .2 SECONDS           'Wait for placement
OUT POB3 = 0                  'Turn actuator back off
FOLM ENABLE NO * * *         'Exit the MPS, does not stop
                               'motor
MOVE 0 0 * *                  'Move XY stage back to home
                               'position, motor now at rest
GOTO PLACE_PT                 'Repeat the cycle
DONE                           'End of program

```

UNIT MASTER

Name	UNIT MASTER					
Descriptor	Set Master Unit Scale Factor					
Type	Set-Up					
Initial value	25000					
Default	UNIT MASTER * * * *					
Syntax	UNIT MASTER 4000 Q23 * 25000					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The **UNIT MASTER** statement, used with the **Following** option, allows master positions to be programmed in user units. In most cases, the user units for master and slave will be the same since they refer to the same physical system, (i.e., inches or revolutions) but this is not required.

The number in the first column of the **UNIT MASTER** statement does not necessarily represent the steps/unit for the encoder input for axis #1. Instead, it represents the steps/unit for whichever encoder input or step output will be the master for the slave on axis 1. This is shown in the example below.

Fractional values are allowed, but truncation errors may occur if the product of the scale factor and the value scaled is not a whole number. Numeric **Q** variables can also be used as the scale factor, and the default value of **UNIT MASTER** is 1. The **UNIT MASTER** parameter scales the following statements: master parameter of **FOL RATIO**, **FOL MDIST**, **FOL MAS_CYC**, **FOL WAIT**, **FOL MOVEWT**, **FOL WIN_P**, **FOL WIN_W**, **FOL M_SYNC**, **FOL S_SYNC**, **IN Qn = FOL AXISn MAS_P** and master parameter of **FOLM RATIO**. After scaling, the result must not be larger than 99,999,999 or an execution error will result.

See Also: **FOL**, **FOLM**

Example

In the example below, axis 1 is following the encoder input on axis #3. Unit scale factors, master and slaves axes, and the following ratio are set.

Statement	Description
UNIT POS 25000 * * *	'Sets slave scale factor to 25000 for axis 1
UNIT MASTER 4000 * * *	'Sets master scale factor to 4000 for axis 1
FOL MASTER ENC3 * * *	'Axis 1 using encoder input #3 as master
FOL RATIO 1:1 * * *	'Set the slave to master ratio at 1 to 1 'based on user units. Actual ratio in steps '= 1*25000 to 1*4000 or 6.25 slave steps for 'each master step.

IN FOL

Name	IN FOL					
Descriptor	Get Following Axis Information					
Type	Programming					
Initial value	N/A					
Range	N/A					
Default	IN Qn = FOL AXISn MAS_P					
Syntax	IN Q10 = FOL AXIS4 MAS_P IN Q1 = FOL AXIS2 TRIG1					
Options	TAB	Q	MAS_P	MAS_C	SLV_P	ETC
	TAB	SHIFT	SYN_ERR	FOL_ERR	TRIG	ETC
	TAB	S_SYNC	M_SYNC	MAS_V		ETC
	F1	F2	F3	F4	F5	F6

Description

- MAS_P** The **IN Qn = FOL AXIS MAS_P** statement assigns to the Q variable the current master cycle position for the following axis specified. It is important to remember that this is not a position of that slave, but instead it is the position of that slave's master. The position returned is the position of the master within its current cycle. For a complete discussion of master cycles, please refer to the section titled *The Master Cycle Concept* earlier in this chapter. The master cycle position in master steps is inversely scaled by **UNIT MASTER** for the axis, so the resulting value in the variable is the cycle position expressed in the user's units. This value may be used for subsequent decision making, or simply recording the cycle position corresponding to some other event.
See Also: **FOL MAS_CYC, FOL NEWCYC, FOL CYC_OFF, IN Qn=FOL AXISn MAS_C**
- MAS_C** The **IN Qn = FOL AXIS MAS_C** statement assigns to the Q variable the current master cycle number for the following axis specified. It is important to remember that this is not a position of the master (or the slave), but instead it is the current cycle number. The master cycle number is set to zero when a new cycle is defined, and is incremented each time a master cycle finishes, i.e., rollover occurs. For a complete discussion of master cycles, please refer to the section titled *The Master Cycle Concept* earlier in this chapter. The master cycle number is not a unit of position, and has no scaling factor. The reported master cycle number is the number of completed master cycles since the last new cycle definition. It will often correspond to the number of complete parts in a production run. This value may be used for subsequent decision making, or simply recording the cycle number corresponding to some other event.
See Also: **FOL MAS_CYC, FOL NEWCYC, FOL CYC_OFF, IN Qn=FOL AXISn MAS_P**
- SLV_P** The **IN Qn = FOL AXIS SLV_P** statement assigns to the Q variable the current slave absolute position for the following axis specified. The position returned will depend on the positioning mode (motor or encoder) of the slave and whether or not the slave is currently in the Moving Positioning System (**FOLM ENABLE YES**). Please refer to the section titled *Moving Positioning System* earlier in this chapter. If the slave is not currently in the Moving Positioning System, the reported position will be identical to that reported with the **IN POS MABS** or **EABS** statement, for motor and encoder positioning respectively. If the slave *is* currently in the Moving Positioning System, the reported position will be the absolute motor or encoder position with respect to the moving positioning system zero reference, for motor and encoder positioning respectively. In other words, this statement may be used to obtain the position of the slave with respect to a moving object or a stationary reference. The position in slave steps is inversely scaled by **UNIT POS** for the axis, so the resulting value in the variable is the slave position expressed in the user's units. This value may be used for subsequent decision making, or simply recording the slave position corresponding to some other event.
See Also: **FOLM DEF, FOLM ENABLE, FOLM PDEF**

SHIFT The **IN Qn = FOL AXIS SHIFT** statement assigns to the Q variable the current value of the net, or absolute slave shift which has occurred at the current ratio for the following axis specified. This ratio may zero, i.e., the slave is at rest, or its current constant ratio reached as a result of a **MOVE SLEWCW** or **MOVE SLEWCCW**. The position returned will be the sum of all shifts performed on that axis since that axis reached constant ratio. Each time a new commanded constant ratio is reached, and at the end of preset distance move, the shift value is set to zero. The reported position will be the net motor or encoder shift for motor and encoder positioning respectively. The shift value in slave steps is inversely scaled by **UNIT POS** for the axis, so the resulting value in the variable is the slave shift expressed in the user's units. This value may be used for subsequent decision making, or simply recording the slave's net shift corresponding to some other event.
See Also: **FOL SHIFT**

FOL_ERR The **IN Qn = FOL AXIS FOL_ERR** statement assigns to the Q variable the current slave following error for the following axis specified. The following error is defined as the difference between the setpoint position and the actual position. For a complete discussion of the conditions which may result in following error, please refer to the section titled *Following Performance* earlier in this chapter. The error value returned will be the motor or encoder position error for motor and encoder positioning respectively. The error in slave steps is inversely scaled by **UNIT POS** for the axis, so the resulting value in the variable is the slave following error expressed in the user's units. This value may be used for subsequent decision making, or simply recording the slave following error corresponding to some other event.
See Also: **FOL MAXACC**, **FOL MAXVEL**

SYNC_ERR The **IN Qn = FOL AXIS SYNC_ERR** statement assigns to the Q variable the current slave synchronization error for the following axis specified. The 4000 allows the user to define two external events, or "marks", which capture the slave position. These are called *Master Sync Mark* and *Slave Sync Mark*, and are defined with the **FOL M_SYNC** and **FOL S_SYNC** statements respectively. Each time either a master or slave sync mark occurs, the corresponding slave position is captured and saved internally. The **FOL SYNC_OFF** statement defines the expected difference between these captured slave positions. This expected difference is called the *Slave Synchronization Offset*. The difference between the actual offset and the expected offset is called the Sync Error. This error may be read into a Q variable using the **IN Qn = FOL AXISn SYNC_ERR** statement. To understand exactly how to use this, please refer to the section titled *Periodic Master/Slave Synchronization* earlier in this chapter.
See Also: **FOL MSYNC**, **FOL SSYNC**, **FOL SYNC_OFF**

TRIGn This is useful in determining whether or not the trigger input has occurred, the first 4 bits in this table will be set when the trigger is defined for a function, then cleared when the trigger occurs. The latter two will be set if the trigger defines the corresponding sync mark, and will remain set until that sync mark is re-defined or defined with a NO parameter. Suppose for example that the trigger had been defined to start a new master cycle and define a master sync mark. Before the trigger occurs, the value assigned to the Q variable will be 17. After the trigger occurs, the value will be 16.

Defines Moving Positioning System	1
Defines new master cycle start	2
FOL WAIT on this trigger	4
FOL MOVEWT on this trigger	8
Defined as master sync mark	16
Defined as slave sync mark	32

This is useful in determining whether or not the trigger input has occurred, the first four bits in this table will be set when the trigger is defined for a function, then cleared when the trigger occurs. The latter two will be set if the trigger defines the corresponding sync mark, and will remain set until that sync mark is redefined or defined with a NO parameter. Suppose for example that the trigger had been defined to start a new master cycle and define a master sync mark. Before the trigger occurs, the value assigned to the Q variable will be 17. After the trigger occurs the value will be 16.

This trigger status request may also be useful by allowing one program to determine if another is waiting on a **FOL WAIT TRIGn** statement when both are running under multi-tasking. The example below shows how to check for only this condition.

```

IN Q1 = FOL AXIS2 TRIG1      'check axis 2's TRIG1 status
MATH Q1 = Q1 AND 4          'check only the FOL WAIT bit
IF Q1 = 4 GOTO IT_WAITS     'if the value is 4, the other
                             'task is waiting

```

S_SYNC The `IN Qn = FOL AXISn S_SYNC` statement assigns the Q variable the slave position most recently latched by the slave synchronization mark defined with the `FOL S_SYNC` statement. Each time the mark occurs, the slave position is captured and may be read with this statement. The position in slave steps is inversely scaled by `UNIT POS` for the axis, so the resulting value in the variable is the captured slave position expressed in the user's units. This value may then be used for whatever purpose is required, even if no master synchronization mark has been defined. These uses may include functions related to Master/Slave Synchronization, or simply general purpose data acquisition. For a complete discussion of Master/Slave Synchronization, refer to *Master/Slave Synchronization*.

M_SYNC The `IN Qn = FOL AXISn M_SYNC` statement assigns the Q variable the slave position most recently latched by the master synchronization mark defined with the `FOL M_SYNC` statement. Each time the mark occurs, the slave position is captured and may be read with this statement. The position in slave steps is inversely scaled by `UNIT POS` for the axis, so the resulting value in the variable is the captured slave position expressed in the user's units. This value may then be used for whatever purpose is required, even if no slave synchronization mark has been defined. These uses may include functions related to Master/Slave Synchronization, or simply general purpose data acquisition. For a complete discussion of Master/Slave Synchronization, refer to *Master/Slave Synchronization*.

MAS_V The `IN Qn = FOL AXISn MAS_V` statement assigns the Q variable the current master velocity. The velocity in master steps per second is inversely scaled by `UNIT MASTER` for the axis, so the resulting value in the variable is the master velocity expressed in the user's units. This value may then be used for whatever purpose is required, such as estimating a process cycle rate. The value returned is only approximate, and its accuracy is dependent on the value chosen for `FOL SMOOTH`. For a complete discussion of the effect of velocity smoothing on velocity measurement accuracy, refer to *Velocity Smoothing*.

ON FOL_ERR

Name	ON FOL_ERR					
Descriptor	Interrupt on Excess Slave Following Error					
Type	Programming					
Initial value	N/A					
Range	N/A					
Default	ON FOL_ERR ANY_AXIS GOTO LABELØ					
Syntax	ON FOL_ERR AXIS2 GOTO F_ERR2					
Options	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The `ON FOL_ERR` statement enables the 4000 to continuously check for excess following error on the specified axis or all axes. For a complete discussion of the conditions which may result in following error, please refer to the section titled *Following Performance* earlier in this chapter. Whenever the slave's setpoint position is not equal to the actual slave position, a following error exists. The user may specify a following error tolerance using the `FOL PTOL` statement. If the magnitude of the actual following error ever exceeds the specified tolerance, the 4000 latches the condition of *following error tolerance exceeded*, and the 4000 will branch to the location specified in the `ON FOL_ERR` statement. If the user's program requires that the 4000 respond to a new occurrence of excess following error, the `FOL PTOL` statement should first be executed to clear the old error, and then the `ON FOL_ERR` statement executed to allow detection of the condition. For a complete discussion of related topics, please refer to the section titled *Monitoring Following Error* earlier in this chapter.

See Also: `FOL PTOL`, `FOL WIN_P`, `FOL WIN_W`, `IN Qn = FOL AXISn FOL_ERR`

ON WT_ERR

Name	ON WT_ERR					
Descriptor	Interrupt on FOL MOVEWT Position Missed					
Type	Programming					
Initial value	N/A					
Range	N/A					
Default	ON WT_ERR ANY_AXIS GOTO LABELØ					
Syntax	ON WT_ERR AXIS2 GOTO WT_QUIT					
Options	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The ON WT_ERR statement enables the 4000 to continuously check for a WT_ERR condition (wait error) on the specified axis or all axes. If a FOL MOVEWT has specified a master cycle position as the wait condition for a subsequent move, and the master position has already exceeded that specified cycle position by the time the subsequent move is commanded, the WT_ERR condition is flagged and latched. For a complete understanding of wait errors, it is important to understand master cycles and waiting for cycle positions. Please refer to the sections titled *The Master Cycle Concept* and *Following Wait Statements* earlier in this chapter. When the WT_ERR condition is detected, the 4000 will branch to the location specified in the ON WT_ERR statement. If the user's program requires that the 4000 respond to a new occurrence of wait error, the FOL MOVEWT statement should first be executed to clear the old error, and then the ON FOL_ERR statement executed to allow detection of the condition. Although it is valid, it is not necessary to specify another wait condition in this case. A FOL MOVEWT NO statement may be used to clear the error.

See Also: FOL PTOL, FOL WIN_P, ON FOL_ERR, IN Qn = FOL AXISn FOL_ERR

DEFINE TRIGDB

Name	DEFINE TRIGDB					
Descriptor	Define trigger debounce time					
Type	Set-Up					
Initial value	80 milliseconds					
Range	4 - 1000 milliseconds					
Default	DEFINE TRIGDB * * * *					
Syntax	DEFINE TRIGDB 80 * Q1					
Options	TAB	Q	NULL			
	F1	F2	F3	F4	F5	F6

Description

The DEFINE TRIGDB statement defines the total debounce time for the four trigger inputs. The debounce prevents noise or the mechanical switch bounce from causing a false interrupt on the trigger input. A trigger is initially recognized on the rising edge of the input. That trigger will not be recognized again until it has gone low and the debounce time, measured from the rising edge, has been exceeded. This debounce time affects registration and all of the FOL and FOLM statements that include triggers as a parameter. The initial value of 80 ms. will usually be long enough to debounce most mechanical and electronic switches, but this time may be lengthened if needed. In some applications, registration marks or master/slave synchronization marks may occur more frequently than 80 ms. In these cases, the debounce time may be shortened, provided the signal *bounce* is short enough. The debounce times are only accurate to ± 2 ms of the specified value, and the actual values used will always be between 4 and 1000 ms. The debounce times are specified for triggers 1, 2, 3, and 4 (left to right on the statement line).

See Also: FOL MOVEWT, FOL NEWCYC, FOL WAIT, FOL M_SYNC, FOL S_SYNC, FOLM DEF

Error Codes

The following is a list of error codes unique to the Model 4000.

Error Messages	Description
Big RAM not installed in U14, U15	This occurs if any FOL or FOLM statement or any IN Qn request for a following parameter is executed without the RAM required for the following option installed.
Invalid FOL MASTER specified	This indicates that an illegal master was specified in FOL MASTER . A slave may never use its own motor step count as its master. A slave in encoder step mode or with stall detect enabled may not use its own encoder step count as master.
FOL MASTER invalid if moving or FOL ENABLE invalid if moving	This indicates the statement is not allowed while the slave is moving. <i>Moving</i> means moving with respect to the current positioning system. A slave may be stationary with respect to a stationary reference, yet be <i>moving</i> in the moving positioning system. FOL MASTER or FOL ENABLE respectively
FOL MASTER not executed	This indicates that no FOL MASTER for the axis is currently specified. It will occur if any FOL or FOLM statement defining or enabling parameters or any IN Qn request for a following parameter is executed and no FOL MASTER statement was executed, or FOL MASTER NO was executed.
FOLM DEF not completed	This indicates that the statement is not allowed if no moving positioning system is defined. It could occur if FOLM DEF was never executed, or if the trigger which defines the moving positioning system has not occurred. FOLM ENABLE YES
FOL parameter too large	This indicates that the numeric parameter supplied with the statement is too large. FOL MAS_CYC —Error if: master steps>99999999 FOL WIN_P —Error if: master steps>99999999 FOL WIN_W —Error if: master steps>99999999 FOL CYC_OFF —Error if: master steps>99999999 or <-99999999 FOL PDEF —Error if: slave steps>99999999 or <-99999999 FOL SYNC_OFF —Error if: slave steps>99999999 or <-99999999 FOL PTOL —Error if: slave steps>99999999 FOL MOVEWT —Error if: master steps >99999999 or <-99999999 FOL WAIT —Error if: master steps >99999999 or <-99999999 FOL WAIT —Error if: master steps >99999999 or <-99999999 FOL M_SYNC —Error if: master steps>99999999 FOL S_SYNC —Error if: master steps>99999999
FOL parameter not valid	This indicates that the parameter supplied with the statement is not valid. FOL MAS_CYC —Error if: master steps are negative FOL RATIO —Error if: ratio denominator is negative FOLM RATIO —Error if: ratio denominator is negative FOL MAXACC —Error if: Error if value is Ø FOL SMOOTH —Error if: smooth number is not 1-4 FOL WIN_P —Error if: master steps are negative FOL WIN_W —Error if: master steps are negative FOL PTOL —Error if: slave steps are negative FOL M_SYNC —Error if: slave steps are negative FOL S_SYNC —Error if: slave steps are negative
Master cycle definition pending	This indicates a master cycle definition is pending a trigger, the master cycle position is unknown. IN Qn AXISn MAS_P IN Qn AXISn MAS_C
FOL SHIFT cannot start move	This indicates that a command phase shift cannot be performed. FOL SHIFT # - Error is already shifting or performing other time based move or VEL or ACCEL is zero or distance is >99999999 or <-99999999 FOL SHIFT CW,CCW - Error if ACCEL is zero
Master sync mark undefined	This indicates that no master sync mark definition exists. This may be because the FOL M_SYNC statement was never executed, or was executed with NO as the parameter. IN Qn AXISn SYN_ERR
Slave sync mark undefined	Indicates that no slave sync mark definition exists. This may be because the FOL S_SYNC statement was never executed, or was executed with NO as the parameter. IN Qn AXISn SYN_ERR
FOL RATIO value invalid	This indicates that the ratio given after scaling by UNIT POS and UNIT MASTER was outside the range ±127, or that the ratio denominator was zero FOL RATIO FOLM RATIO

I N D E X

A

ABSOLUTE COORDINATE SYSTEM 40
ABSOLUTE ENDPOINT PROGRAMMING 40

C

CIRCLES 21
COORDINATE SYSTEMS 16, 40
CURRENT ERROR 75
CUSTOM PRODUCTS GROUP 11

D

DEVICE CLEAR, INTERFACE CLEAR 6
DISTANCE CALCULATIONS 108
DYNAMIC POSITION MAINTENANCE 103

E

ELECTRONIC GEARBOX 67
ERROR
 POSITION 104
 SYNC 78, 82
 TOLERANCE 129

F

FILTERING 103
FOL SMOOTH 62
FOLLOWING ERROR 75, 128, 129, 130, 138, 139
FOLLOWING MODE 65, 66, 111, 117

H

HOST COMPUTER PROGRAMS 5

I

IEEE-488
 COMMUNICATION 5
 INSTALLATION 1
INCREMENTAL PROGRAMMING 16
INSTALLATION 62

L

LOCAL COORDINATE SYSTEM 16, 40

M

MASTER
 CYCLE 128
 DISTANCE 65, 66
 SYNC MARK 77, 78, 80, 82, 126, 138
 VELOCITY 62, 102, 103
MASTER CYCLE
 CONCEPT 69
 LENGTH 69, 70, 120, 122, 129
 NUMBER 69, 137
 POSITION 69, 70, 75, 78, 96, 119, 122, 126, 137, 140
MONITORING FOLLOWING ERROR 74
MOTION
 ROUGH 103
MOTION PATHS 11
MOVING POSITIONING SYSTEM 61, 94, 110, 132, 133, 134
MULTI-TASKING EXAMPLE 54

N

NON-PATH ACTIONS 27
NON-PATH STATEMENTS 26

O

OFFSET

ACTUAL 77, 128, 138
EXPECTED 77, 78, 128, 138
SYNCHRONIZATION 77, 79, 80, 82, 128, 138

P

PATH
 ACCELERATION 16
 COMPILING 26
 DECELERATION 16
 DEFINITION 14
 EXECUTING 26
 EXECUTION 14
 LENGTH 18
 ORIENTATION 18
 PLACEMENT 18
 VELOCITY 16
POB OUTPUT 23, 28, 31
POSITION
 ERROR 74
 INITIAL 95, 132
 MASTER CYCLE 75, 78
 RELATIONSHIP 66
 SETPOINT 74, 102, 139
PROGRAMMING
 ABSOLUTE 17, 40
 ERRORS 26
 EXAMPLES 28
 INCREMENTAL 17, 40

R

RAMP 66
RANDOM TIMING INFEED 77, 79
RATIO 65
RATIO FOLLOWING 61, 64, 68, 94
REGISTRATION 81, 111
ROLLOVER 69, 78
ROUGH
 MOTION 103
 MOTION IS
 MOTION 111

S

SEGMENT BOUNDARY 21
SERIAL POLL REGISTER 4, 5
SETPOINT
 POSITION 74, 102, 139
SHIFT 65, 81, 82
SLAVE 65
 POSITION 66
 SHIFT 66
 SYNC MARK 77, 78, 80, 82, 127, 138
STATEMENT
 ACCEL 66, 108, 112
 ACCEL PATH 32
 DECEL PATH 32
 DEFINE GPIB ADDR 8
 DEFINE GPIB ERR_MSG 8
 DEFINE GPIB PROMPTS 8
 DEFINE GPIB SRQ 5
 DEFINE GPIB SRQ IF SPAS 7
 DEFINE ON RET 58
 DEFINE TRIGDB 59, 140
 DISPLAY ON PORTN TRACE 57
 ENABLE REGSRV 59
 ENABLE TRIG REV 58
 FOL CAM 131
 FOL CYC_OFF 70, 126

FOL DIRSET 130
 FOL ENABLE 65, 117
 FOL ENCCCHK 131
 FOL LEAD 130
 FOL MASTER 115
 FOL MAS_CYC 121
 FOL MAXACC 74, 112, 124
 FOL MAXVEL 74, 112, 125
 FOL MDIST 65, 66, 108, 112, 118
 FOL MOVEWT 71, 119, 140
 FOL M_SYNC 77, 126
 FOL NEWCYC 69, 121
 FOL PTOL 75, 112, 129
 FOL RATIO 65, 66, 117
 FOL SHIFT 65, 78, 83, 116
 FOL SMOOTH 73, 124
 FOL SYNC_OFF 77, 79, 128
 FOL S_SYNC 77, 127
 FOL VELFF 73, 102, 125
 FOL WAIT 122
 FOL WIN_P 75, 128
 FOL WIN_W 75, 129
 FOLLOWING STATEMENT GROUP
 FOL MAS_CYC 69
 FOL WAIT 70
 FOLLOWING PERFORMANCE 73
 MASTER CYCLE 71
 MOVING POSITIONING SYSTEM 94, 96
 RATIO FOLLOWING 64
 SUMMARY OF FOLLOWING PERFORMANCE AND
 MEASUREMENT STATEMENTS 76
 SUMMARY OF PERIODIC MASTER/SLAVE
 SYNCHRONIZATION STATEMENTS 79
 SUMMARY OF RATIO FOLLOWING STATEMENTS 67
 FOLM DEF 94, 133
 FOLM ENABLE 95, 134
 FOLM PDEF 94, 132
 FOLM RATIO 94, 132
 IN FOL 137
 FOL MAS_CYC 70
 FOL_ERR 138
 MAS_C 137
 MAS_P 70, 137
 SHIFT 138
 SLV_P 66, 95, 137
 STATEMENTS. 78
 SYNC_ERR 77, 138
 MAS_V 139
 M_SYNC 139
 ON FOL_ERR 75, 130, 139
 ON WT_ERR 140
 OUT SPOL 9
 PATH 27, 33
 PATH COMPILE 24, 36
 PATH C_RES 44
 PATH DEF 34
 PATH END 35
 PATH EXECUTE 24, 34
 PATH LINE 39
 PATH OCCW 39
 PATH OCW 38
 PATH ONLY 42
 PATH POB 43
 PATH P_RATIO 44
 PATH RAD_TOL 46
 PATH RCCW 37
 PATH RCW 37
 PATH UNCOMP 14, 45
 PATH XY 40
 S_SYNC 139
 TASK 56
 TRIGN 138
 UNIT MASTER 64, 65, 136
 UNIT PATH ACCEL 49
 UNIT PATH POS 47
 UNIT PATH VEL 48
 UNIT POS 64, 65
 VEL PATH 50
 WAIT 71
 WT_ERR 140
 SUBROUTINES 29
 SYNC ERROR 128, 138
 SYNC MARK 78
 SYNCHRONIZATION 27, 69, 71, 75, 77, 126, 127
 ERROR 78, 138
 SYNCHRONIZING 71
 SYNC_ERR 78, 79, 138

T

TRACKBALL 68
 TRIGGER 70, 71, 78, 94, 119, 122, 126, 127, 133

V

VELOCITY
 CORRECTION 104
 SMOOTHING 62
 VELOCITY AVERAGING 74, 103, 124
 VELOCITY FEED FORWARD 62, 74, 102, 103
 VELOCITY SMOOTHING 103

W

WAITS 69
 WEB PROCESSING 77, 81
 WINDOW
 ERROR DETECTION 75, 128, 129
 START POSITION 128
 STARTING POSITION 75, 76
 WIDTH 75, 129
 WORK COORDINATE SYSTEM 16, 29, 40