

Servida Motor Version Update

Motion Control Chip Optional Firmware

Version 4.76

for SVxxxx-PLS series

This document covers features of the release of new firmware version 4.76.

Version 4.76 Firmware has substantially increased functionality and has some differences with the standard firmware.

Please refer to the Summary of Changes on page 3 for a list of differences between Version 4.76 and 4.15 /4.40 firmware.

To find out what Servida Motor firmware version you currently have, type "**RSP**" command in the "Servida Motor Terminal" window in the Servida Motor Interface software or any ASCII terminal program. The motor responds by returning the sample rate and firmware version.

For example: After issuing RSP if you get a response of "**24576/476**" the numbers after the slash "/" represent the firmware version number. In this example it is a 4.76 firmware version motor.

This document is intended to serve as an addendum to the existing manual.

TABLE OF CONTENTS

INTRODUCTION	1
TABLE OF CONTENTS	2
SECTION 1.0 SUMMARY OF CHANGES	3
1.1 Features Added	3
1.2 Commands Added	4
1.3 Commands Removed	5
1.4 Commands Modified	5
SECTION 2.0 BOOT-UP DEFAULTS	6
SECTION 3.0 LIMITS	6
3.1 Hardware Limits	6
3.2 Software Limits	7
SECTION 4.0 FAULT HANDLING	8
4.1 Fault Handling Sequence	9
4.2 Clearing Fault	9
4.3 Fault Handling Commands	9
SECTION 5.0 BRAKE FUNCTIONS	11
5.1 Internal/External Brake Commands	11
5.2 Dynamic Braking, Mode Torque Brake	12
SECTION 6.0 INPUTS/OUTPUTS	13
6.1 I/O Commands Added	13
6.2 Reporting Digital I/O	13
6.3 Reporting Analog I/O	13
SECTION 7.0: F= FUNCTION COMMANDS	14
SECTION 8.0: VARIABLES	17
8.1 Forced Equal Sign	17
8.2 Arrays	17
SECTION 9.0 : MODE CAM ADDITIONAL FUNCTIONS	18
9.1 Definitions	18
9.2 Cam Modulo Mode	18
9.3 Relative Positioning Mode in Cam Mode	20
9.4 Mode Cam Limitations	22
9.5 On-the-Fly Parameter Changes in Cam Mode	22
SECTION 10.0 : HOST MODE	24
10.1 Automatic Strobing of Host Mode Time	24
10.2 Orderly Host Mode Exit	24

1.0 SUMMARY OF CHANGES

Version 4.76 firmware has added many features and has made changes to existing 4.15B, 4.15C and 4.40 firmware. The differences are summarized below.

1.1 Features Added

Soft Limits	Software set positional limits.
Hard Limits	Hardware limits are active-high asserted and Directional only. Only normally closed limits, dry contacts or sinking (NPN) switches can be used with version 4.76.
Fault Handling	Upon any shaft protection motor fault the running program will END. There is an option to call subroutine C1 on fault interrupt.
Improved I/O Handling	Read all seven I/O pins as 8 bit value variable "U". Read individual I/O using single report command. Print I/O as individual or seven I/O pin with single command.
I/O Pin Initiated Gosub	I/O pin G can be configured to call subroutine C2 on interrupt when triggered.
External Brake Control	Output I/O pins C or G can be configured to control an external brake on fault interrupt or during trajectory.
Dynamic Braking	Mode torque brake (MTB) provides dynamic braking by shorting the coils to provide a braking torque. Note: This is the default mode on power-up and MTB is engaged on any shaft protection fault.
Dwell in Mode Cam	Remain at the last position of a cam table for a specified number of external encoder counts. Dwell is accessible in all cam modes.
Relative Mode Cam	Allows cam table to have non-zero end point. Motor moves relative from that point into next cam cycle.
Cam Modulo	"Anti-wrap" Resets both internal position and external counter to zero at end of each cam cycle. The motor shaft position and Master Encoder counter is relative to the start of each cycle.
Improved Host Mode	Reduced the information sent in order to increase bandwidth and provides an orderly host mode exit.
Required "="	All parameter and variable assignments require the "=" sign. For example: A100 is no longer accepted. A=100 must be used. Multiple setting of arrays is still permitted. For example: aw[0] 0 100 500 700. which is equivalent to: aw[0]=0 aw[0]=100 aw[0]=500 aw[0]=700

1.2 Commands Added

BRKC	Brake output signal is active low on user pin C and the internal brake pin when preceded by command UCO . (Brake control follows BRKSRV or BRKTRJ commands accordingly).
BRKG	Brake output signal is active low on user pin G and the internal brake pin when preceded by command UGO . (Brake control follows BRKSRV or BRKTRJ commands accordingly).
BRKI	Brake output signal is on internal brake pin only. This is the default. NOTE: BRKG , BRKC , and BRKI are mutually exclusive.
CI	Re-initializes any parameter changes in cam mode at next zero cross-over.
CX	Value of current cam index in mode cam.
F=2	Reverse the directions of positive/negative shaft rotation. Also see section 1.4 modified commands.
F=16	Cause cam mode to operate in relative position mode.
F=32	Enables Interrupt call to C1 on a new occurrence of motor protection fault. Faults include: Position Error, Software or Hardware Limits, Continuous Over Current, Over Temperature
F=64	Enables Interrupt call to C2 on a user pin G high to low edge transition.
F=128	Cause Internal position, (@P or RP), and external counter, CTR , to be reset at modulo cam BASE plus dwell (D).
MTB	Immediately set mode torque brake.
RETURNF	Return from Interrupt-call to C1 , (See F=32 above).
RETURNI	Return from Interrupt-call to C2 , (See F=64 above).
RU	Report current logic level state of user pins A through G as a 7-bit binary value.
RU{pin}	Report the digital value of the user I/O pins (input or output). Value 1 or 0 reported. The commands are: RUA,RUB,RUC,RUD,RUE,RUF,RUG .
RU{pin }A	Report the analog to digital converted value of the user pin (input). Value 0 to 1023, where 0 is 0 volts and 1023 is 5.0 volts. The commands are: RUAA, RUBA, RUCA, RUDA, RUE, RUFA, RUGA .

SLD Disable software limits.

SLE Enable software limits.

SLN=<expression>

Set negative software limit. Values must be between -2,147,483,648 to 2,147,483,647.

SLP=<expression>

Set positive software limit. Values must be between - 2,147,483,648 to 2,147,483,647.

U 7-bit value of user input/output pins A through G.

Ze Reset to zero position error flag bit.

Zh Reset to zero historical overheat flag bit.

1.3 Commands Removed

ES400
ES1000
KGON
KGOFF
LIMH
LIML
LIMN
MFRATIO
RES=
MD50

1.4 Commands Modified

F=2 Changed function to reverse positive orientation of shaft. Prior function , F=2, permit scan error host output transmissions.

2.0 BOOT-UP DEFAULTS

Version 4.76 Servida Motors have additional error handling code that further protects both the user and the Servida Motor from damage.

These additions are as follows:

1. Hardware limits are active-high asserted and directional only.
With nothing wired to limits, motor will boot-up in fault condition.
2. Default Mode is Mode-Torque Brake. It is also the default mode under any shaft protection fault:
(Position error, Over Temp/RMS over current, Soft/Hard Limits)
3. Default brake control is BRKSRV.
4. Any motor protection fault results in ending the program.
(Position error, Over Temp/RMS over current, Soft/Hard Limits)
5. No further shaft movement can be commanded until prior faults (as listed above) are cleared.
6. Downloading of a program is prevented when in a fault condition.
7. Option for interrupt driven subroutine call on motor error via F command.

3.0 LIMITS

3.1 Hardware Limits

I/O pin C (Positive or Right Limit) and I/O pin D (Negative or Left Limit) are active high asserted and directional only. They require normally closed limit switches pulling the pins to ground. Loss of ground potential results in limit fault. (Limit inputs have 5KOhm pull-up resistors internally so a loss of connection will result in fail-safe limit fault.)

As a result of above said conditions: on power-up with no program and nothing wired to I/O pin C and D, Version 4.76 will see this as both Positive and Negative Limits asserted and will be in an error state. No motor shaft movement can be commanded until Limit switch errors are cleared.

In order to clear Hardware Limit Switch Input Errors, you must do the following:

1. Issuing **UCI** and/or **UDI** to reassign the limits as general inputs
2. or grounding Ports C and/or D,
3. and then clearing the Left and Right limit status bits via
 - a. **ZS** Reset all system state flags **or**
 - b. **ZI** Reset Historical left limit (Port C) **and/or**
 - c. **Zr** Reset Historical right limit (Port D)

NOTES

1. You can move in the opposite direction of approach to move off of the Limit, but the Historical Limit Status bit must be cleared first.
2. There are no provisions to make the hardware Limits Active Low or Non-Directional in Version 4.76.
3. Sample code to place at the top of a Version 4.76 firmware Servida Motor:

```
UCI 'assign right limit to be used as standard user-input port
UDI 'assign left limit to be used as standard user-input port
ZS 'clear all status word, including limit and position error bits
```

...Continue on with normal code...

3.2 Software Limits

Only Version 4.76 firmware or greater has software limits. When enabled, the CPU continuously monitors shaft position. If the shaft reaches the negative soft limit value (SLN) or positive soft limit value (SLP) the motor behaves as if a hardware limit was triggered.

NOTE:
By Default, the motor shaft will rapidly stop via **MTB**<Mode Torque Brake>.

The commands are:

SLE <Software Limit Enable>: Enables Software Limits.

SLD <Software Limit Disable>: Software Limits are disabled by Default on Power-up.

SLP=<expression>

Software Limit Positive Variable (Default value is 0).

Set positive soft limit value to expression to the right of equal sign.

SLN=<expression>

Software Limit Negative Variable (Default value is 0).

Set negative soft limit value to expression to the right of equal sign.

NOTE:
Software Limits must be disabled via the SLD command before their values are changed.
Their values are not affected by the command **O**=expression, but measured position is affected.

Upon reaching a Limit (mechanical protection fault) the controller response is:

1. Turn off servo amplifier, if F=1 is not issued.
If F=1 is issued the motor decelerates to a stop at the acceleration value set by "A".
2. Apply dynamic brake, MTB.
3. Send signal to engage static brake, external and/or internal, if motor is so equipped.
4. Program execution continues to run.

NOTE:

1. During a mechanical protection fault, the commands **G**, **MT**, **MF1**, **MF4**, **MS**, **S**, **X** and G-synchronization function on I/O pin G is inhibited and no action occurs when these commands are received. The fault has to be cleared before these commands can be re-enabled.
2. You can move in the opposite direction of approach to move off of the limit only after clearing the associated Historical Limit Status.
3. Shaft Stops w/regards to F command:
 1. First bit = 0 (Deceleration to stop disabled):
 - a. With **MTB** enabled, the motor will rapidly and dynamically brake to a stop.
 - b. With **MTB** disabled the motor will freewheel coast to a stop. **MTB** is disabled by issuing commands BRKRLS and OFF in this sequence.
 2. First bit = 1 (Deceleration to stop enabled):
 - a. With **MTB** enabled, the motor will decelerate to a stop and then switch to **MTB**.
 - b. With **MTB** disabled the motor will decelerate to a stop and hold position. **MTB** is disabled by issuing commands BRKRLS and OFF in this sequence.

Note: See section 4.3 for more detail on the “**F**” commands.

4.0 FAULT HANDLING

Version 4.76 has added a fault-handling sequence for improved motor protection. The faults that activate the fault handling sequence are:

Table 4.1: Fault Handling Flags

FAULT DESCRIPTION	STATUS BIT AFFECTED	STATUS BIT #	STATUS BIT VALUE	DESCRIPTION	FAULT TYPE
Positive (Right) Limit Exceeded	Br Historical Positive (Right) Limit	1	2	Position exceeded hard limit (I/O pin C) or soft limit.	Mechanical Protection Fault
Negative (Left) Limit Exceeded	Bl Historical Negative (Left) Limit	2	4	Position exceeded hard limit (I/O pin D) or soft limit.	Mechanical Protection Fault
Position Error Exceeded	Be Historical Position Error	5	32	Position error exceeded error limit.	Motor Protection Fault
Overheat	Bh Historical Overheat	6	64	Controller temperature exceeded temperature limit or RMS current exceeded limit.	Motor Protection Fault

4.1 Fault Handling Sequence

Upon a mechanical protection fault the controller response is:

1. Turns off servo amplifier, if F=1 is not issued.
If F=1 is issued, the controller decelerates to a stop and then servos in place.
2. Apply dynamic brake, MTB, if MTB function is not de-activated.
3. Send signal to engage static brake, external and/or internal, if function is activated.
4. Program execution continues to run, if F=32 is not issued.
If F=32 is issued, program goes to label "C1".

NOTE: During a mechanical protection fault, the commands G, MT, MF1, MF4, MS, S, X and G-synchronization function on I/O pin G is inhibited and no action occurs when these commands are received. The fault has to be cleared before these commands can be re-enabled.

Upon a motor protection fault the controller response is:

- 1 Turn off servo amplifier, if F=1 is not issued.
If F=1 is issued, the controller decelerates to a stop then servos in place.
- 2 Apply dynamic brake, MTB, if MTB function is not de-activated.
- 3 Send signal to engage static brake, external and/or internal, if function is activated.
- 4 End program execution, if F=32 is not issued.
If F=32 is issued, program goes to label "C1".
Note, during a motor protection fault the "RUN" command is inhibited.

4.2 Clearing Fault

The fault must be cleared before the motor responds to the inhibited commands. To make the motor respond to these commands the status bits have to reset by ZS or clear the fault bits individually by Zr, Zl, Ze, and Zh.

4.3 Fault Handling Commands

You can modify the fault handling sequence using the new commands:

Table 4.2: "F" Functions

F=0	Deceleration to stop disabled. Upon a software or hard limit, the motor turns off servo amplifier.
F=1	Deceleration to stop enabled. Upon a software or hard limit, the motor decelerates to a stop. * With MTB enabled, the motor will decelerate to a stop and then switch to MTB . * With MTB disabled the motor will decelerate to a stop and hold position. MTB is disabled by issuing commands BRKRLS and OFF in this sequence.
F=32	Program goes to label C1 when a motor protection , error or hot bit, fault occurs and stores it program pointer in the stack. Use the "RETURNF" command at the end of subroutine, to return to program pointer when the input was triggered.
F=64	Go to subroutine label C2 (GOSUB2) when the I/O pin G goes from a high to low edge transition. Use the "RETURNI" command at the end of subroutine, to return to program pointer when the input was triggered.

To use more than one "F=" function add the bit values. For example if you want "F=1" and "F=32", issue command "F=33".

SAMPLE: FAULT HANDLING PROGRAM

MAIN PROGRAM:

```
ZS          ' Reset all status bits
F=33        ' F=32+1 where F=32 goes to subroutine C1 on a fault
            ' F=1 Decelerate to a stop when limit is triggered
MP          ' Set to position control mode
A=8*25      ' Set acceleration
V=537*600  ' Set velocity
P=2000*10  ' Set position
G           ' Go
END         ' End of main program
```

```
-----
C1          ' Fault handling subroutine
  IF Be==1  ' If position error exceeded,
    pp=@PE  ' Store real time position error into variable.
    GOSUB200
  Ze        ' Reset position error bit
ENDIF
RETURNF     'Return to main program where fault was triggered.
*****
C200
  PRINT("POSTION ERROR= ",pp,#13)
RETURN
*****
```

5.0 BRAKE FUNCTIONS

The Servida Motor Version 4.76 controller firmware can be used with an external brake or internal integral static brake. The I/O can be used to control an external static fail-safe brake. Static brakes are used to apply a holding torque when the motor shaft is not moving.

Note 4.15B and 4.15C can only use I/O pin G to provide external brake control.

5.1 Internal/External Brake Commands

BRKI This is the default state. The controller controls the Servida integral brakes using internal control pins and an internal brake power supply.

BRKG Provides a 5Vdc control signal to I/O pin G to control an external brake in addition to controlling the internal brake. The signal is 0 Vdc when brake is released. The I/O pin is set to 5 Vdc when brake engaged.

NOTE:

1. **UGO** must be issued to assign Port G as an Output before the **BRKG** command is issued.
2. When **BRKG** is used, do not issue the following commands:
RS4 or OCHN(RS4,0,...)
UGI
UG=<value>
<variable>=UG

These commands will change the function of the I/O pin G and may cause the brake to release.

3. If **BRKSRV** is issued, Port G acts as a "Motor Fault" output (Driven low when not faulted). If F=32 is issued, C1 will also be called.
4. If **BRKTRJ** is issued, Port G acts as a "Busy Trajectory" output (Driven low while trajectory bit is 1). If F=64 is issued, C2 will also be called.

BRKC Provides a 5Vdc control signal to I/O pin C to control an external brake in addition to controlling the internal brake. The signal is 0 Vdc when brake is released. The I/O pin is set to 5 Vdc when brake engaged.

NOTE:

1. **UCO** must be issued to assign Port C as an Output before the **BRKC** command is issued.
2. When the **BRKC** is used, do not issue the following commands:
UCI
UC=<value>
<variable>=UC

These commands will change the function of the I/O pin C and may cause the brake to release.

3. **BRKSRV** is issued, I/O pin C acts as a "Motor Fault" output (Driven low when not faulted).

4. If **BRKTRJ** is issued, I/O pin C acts as a “Busy Trajectory” output (Driven low while trajectory bit is 1).

5.2 Dynamic Braking, Mode Torque Brake

Version 4.76 adds dynamic braking feature used to improve deceleration under a fault condition or when asserted via the **MTB** command. Dynamic braking applies braking torque to a moving motor shaft by internally shorting the motor coils. This creates a resistance torque and helps decelerate the motor.

MTB (Mode Torque Brake) Applies dynamic braking torque. To turn off the dynamic brake (**MTB**) function issue **BRKRLS** <Brake Release> and then **OFF** <Servo Off>. This also prevents **MTB** from automatically being issued upon any shaft protection fault. To re-enable the mode torque brake issue **MTB** again.

NOTE:

1. **MTB** is the mode the motor defaults to under the following conditions:
 - 1A Negative (Left) or Positive (Right) Limit asserted, software or hardware limit.
 - 1B RMS Over current/Over Temperature , both share same Status bit: Bh.
 - 1C Position Error (Be status bit)
2. The **RMODE** command will report “B” when **MTB** is active.
3. **MTB** follows interrupt controls outlined by the **BRKSRV**, **BRKTRJ**, and **BRKRLS** commands which define how a Servida Motor operates under conditions outlined

below.

BRKSRV When Status bit **Bo** (motor-off) is 0, **MTB** will be disabled.
When **Bo** is 1, **MTB** will be active.
Note that when the motor is off for any reason, **MTB** will be active.
Version 4.76 defaults to **BRKSRV** being the active state, as a result, **MTB** is active on power-up.

BRKTRJ When **Bt** (busy-trajectory) is 1, **MTB** will be disabled.
When **Bt** is 0, **MTB** will be active.
MTB will de-activate whenever the motor is working towards a trajectory.

BRKRLS When **BRKRLS** is issued and followed by the **OFF** command, **MTB** will de-activate and stay inactive until the command **MTB** is issued.
Issuing **BRKENG** will not activate **MTB**.

NOTE:

1. **G**, **MT**, **MF1**, **MF2**, **MF4**, and/or **MS** commands will turn off **MTB** and switch to the associated mode.
2. **MTB** will then follow **BRKTRJ** or **BRKSRV** commands while in those modes or at their termination as described above.

6.0 INPUTS/OUTPUTS

Version 4.76 has added commands to improve code writing of I/O handling. The state of all 7 I/O pins can be read as a single 7-bit value in variable "U". The value of U has the bit pattern:

Table 6.1: Input Output Word Value

	Value of U						
BIT#	6	5	4	3	2	1	0
I/O PIN	G	F	E	D	C	B	A
VALUE	64	32	16	8	4	2	1

Report digital value of individual I/O pin as 1 or 0. Where logical 1 is a voltage 2.0 Vdc or greater. Logical 0 is ≤ 0.8 V dc.

6.1 I/O Commands Added

- RU** Print value of all 7 I/O as a digital value 0 to 127 (base 10)
To current communication channel.
- <variable> =U** Store the value of "U" into a user variable. U can be operated by logical and mathematical operations.
- PRINT(U)** Print to communication channel 0 the value of U.
- PRINT1(U)** Print to communication channel 1 (RS485 channel, I/O pin E and F) the value of U. This command is not for use on models which do not have separate RS485 channel on I/O pin E and F.

6.2 Reporting Digital I/O

Reported values are 0 or 1, where 1 is 2.0 volts or higher and 0 is 0.8 volts or lower. These commands prints to the active RS232 or RS485 channel.

- RUA** Report to communication port digital value of User I/O pin A
- RUB** Report to communication port digital value of User I/O pin B
- RUC** Report to communication port digital value of User I/O pin C
- RUD** Report to communication port digital value of User I/O pin D
- RUE** Report to communication port digital value of User I/O pin E
- RUF** Report to communication port digital value of User I/O pin F
- RUG** Report to communication port digital value of User I/O pin G

6.3 Reporting Analog I/O

Reported values are 0 to 1023 using a 10 bit analog to digital converter. Where 1023 is 5.0 Vdc and 0 is zero volts. These commands prints to the active RS232 or RS485 channel.

- RUAA** Report to communication port analog value of User I/O pin A
- RUBA** Report to communication port analog value of User I/O pin B
- RUCA** Report to communication port analog value of User I/O pin C
- RUDA** Report to communication port analog value of User I/O pin D
- RUEA** Report to communication port analog value of User I/O pin E
- RUFA** Report to communication port analog value of User I/O pin F
- RUGA** Report to communication port analog value of User I/O pin G

7.0 F= FUNCTION COMMANDS

Special functions “F=<expression>” commands have been added. Table 6.1 lists all “F=” commands and details the changes. To use more than one F command issue a “F =” sum of “F=” functions value. Example: F=35 activates F=1, F=2, and F=32 functions. Upon power up F=0 is the default state.

Table 7.1: Value of “F” Function Commands

BIT	DESCRIPTION	F Value (Function Enabled)	COMMENT
0	<p>F=0 Sets motor to turn servo amplifier off on a hard or soft limit trigger. Default state.</p> <p>F=1 Upon a software or hard limit, the motor decelerates to a stop. Bo will equal 0. If MTB is disabled and F=0 command is issued, the motor will free wheel on reaching either a soft or hard limit. Set bit =1.</p>	F=1	New function
1	<p>F=2 Reverse shaft direction. Upon issuing the command the motor shaft will physically spin in the opposite direction for any given mode including Mode Torque. This will take effect at the moment the second bit of F is set regardless of what mode you are in.</p>	F=2	Change existing function meaning
2	<p>F=4 Directs user program prints and report command to channel 1 (RS485 channel, I/O pin E & F).</p>	F=4	Existing function
3	<p>F=8 Clear PID integral term (KI) at the end of a trajectory.</p>	F=8	Existing function
4	<p>F=16 Use relative position mode when in cam mode.</p>	F=16	New function
5	<p>F=32 Program goes to label C1 when a motor protection (position error or hot bit) fault occurs and stores it’s program pointer in the stack. Use the “RETURNF” command at the end of subroutine, to return to program pointer when the input was triggered.</p> <p>F=32 causes a GOSUB1 when any of the following motor fault protection bits becomes active (=1): Be: Position Error Bh: Over Temperature (this is also RMS over-current) Bp: Real time Positive Limit via software or hardware limits Bm: Real time Negative Limit via software or hardware limits</p>	F=32	New function

5	<p>F=32 operates independently of F=64 even if F is set to 96:</p> <ol style="list-style-type: none"> 1. If both a fault-triggered GOSUB1 and an Input trigger event to GOSUB2 occur at the same time, The fault takes priority and GOSUB2 will not be called. 2. If an input trigger called GOSUB2. and then a Fault occurs, C1 will be called and return to C2 when the error is cleared. 3. If a port G input trigger occurs while in C1, a call to C2 <u>will</u> occur because C1 has already been called. The Motor will enter back into C1 at the end of C2 if Port G clears. <p>Action taken when a Command is in progress:</p> <ul style="list-style-type: none"> • The G pin is polled once per sample and is de-bounced. Code may be executing or in process when this occurs. • The WAIT command is the <u>only</u> motor instruction that will be truncated automatically when GOSUB2 is called on interrupt. All other valid commands will finish processing prior to the call. <p>Example: WAIT=8000 is issued and port G goes low <4000 clock ticks into the wait period, the WAIT will be terminated. Long PRINT commands will be completed prior to GOSUB2 call on interrupt.</p> <p>Note: F=32 operates independently of F=64. A pin G-triggered GOSUB2 may occur while performing GOSUB1 processing. To inhibit this, the first instruction of routine C1 must be (if the value of F is dynamic and unknown) F=F&191 (191+64=255). Since the value of F will not have been saved, a shadow variable could be used when setting F.</p>	F=32	
6	<p>F=64 Go to subroutine label C2 (GOSUB2) when the I/O pin G goes from a high to low edge transition. Use the "RETURNI" command at the end of subroutine, to return to program pointer when the input was triggered.</p> <p>Note on Commands in progress:</p> <ul style="list-style-type: none"> • The G pin is polled once per sample and is de-bounced. Code may be executing or in process when this occurs. <p>The WAIT command is the <u>only</u> motor instruction that will be truncated automatically when GOSUB2 is called on interrupt. All other valid commands will finish processing prior to the Call.</p>	F=64	New function

6	<p>Example: WAIT=8000 is issued and port G goes low <4000 clock ticks into the wait period, the WAIT will be terminated. Long PRINT commands will be completed prior to GOSUB2 call on interrupt.</p> <p>Re-entrance of GOSUB2 is the responsibility of the user. To avoid the possibility of unintended reentrance or a recursive loop overflowing the stack, a new input trigger occurrence will not trigger a GOSUB2 until a prior GOSUB2 call has been completed by:</p> <ul style="list-style-type: none"> • A return by RETURNI rather than RETURN; • Or STACK (which should immediately be followed by GOTO); • Or, END. <p>F=64 operates independently of UG, UGI, and UGO:</p> <ol style="list-style-type: none"> 1. F=64 and UG: GOSUB2 triggered with concurrent "Go" on input pin G high to low edge transition; 2. F=64 and UGI: GOSUB2 triggered on input pin G high to low edge 3. F=64 and UGO: GOSUB2 triggered on UG=0 (when Port G goes from High to Low). <ol style="list-style-type: none"> a. Note: If BRKG is instated, port G will be controlled by associated brake commands and C2 will be called anytime port G goes low due to brake control. <p>F=64 operates independently of F=32 even if F is set to 96:</p> <ol style="list-style-type: none"> 4. A fault-triggered GOSUB1 may occur while performing GOSUB2 processing. 5. GOSUB1 fault trigger takes priority over GOSUB2. 	F=64	
7	<p>F=128 Resets value of "@P", "RP" and external counter "CTR" to be set to zero at modulo "BASE" plus dwell "D". Relative cam Mode.</p>	F=128	New function

8.0 VARIABLES

8.1 Forced Equal Sign

When setting variables and parameters, the equal “=” sign must be present. For example, A100 is no longer accepted.

8.2 Multiple Arrays

The number of arrays variable type ab[...] and aw[...] has been increased.

1. The array byte variable has increased from ab[200] to ab[203].
2. The array word variable has increased from aw[100] to ab[101].
3. The variables ab[], aw[] and al[] share the same memory.

The array variable map below shows how the variables overlay.

Table 8.1: Array Variable Map

TYPE	FIRST ARRAY LOCATION				LAST ARRAY LOCATION			
ARRAY BYTE ab[]	ab[0]	ab[1]	ab[2]	ab[3]	ab[200]	ab[201]	ab[202]	ab[203]
	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB
ARRAY WORD aw[]	aw [0]		aw[1]		aw[100]		aw[101]	
	MSB		LSB		MSB		LSB	
ARRAY LONG al[]	al[0]				al[50]			

9.0 MODE CAM ADDITIONAL FUNCTIONS

In mode cam the motor position bases its position upon the encoder signal from an external source. The position is defined by a position table, "cam table", specified by array table aw[0] to the maximum of aw[101]. New features **dwll, mode cam modulo (anti-wrap)** and **relative position mode in cam mode** have been added.

9.1 Definitions

Cam Modulo	Set position and counter to zero after the end of each cam table.
CI	"cam Initialize" command added to cause the buffered cam parameters to replace the current cam parameters, including the dwell value.
CX	Variable contains real-time cam index while in cam mode. This enables the user to avoid the active area of the cam table when dynamically modifying the cam table data.
D	Dwell is the number of external encoder counts the motor remains at the last cam table position for specified. Dwell can be set to zero.
F=16	Command added to cause cam MODE to operate in relative position mode. In this mode, the motor moves to positions relative to where it is when it starts into the cam table.
Relative Position Mode In Cam Mode	Set last position of cam table as start of cam table. Cam table is relative to that new zero position.
SIZE=<value>	In relative cam mode and cam modulo, size is the highest array index value, aw[<index>]. For example a program with array locations aw[0] to aw[10] the size is 10.

9.2 CAM MODULO MODE

In cam modulo mode (cam anti-wrap mode) the position and external encoder input counter is reset to zero when the external encoder input reaches the value of cam cycle length, "BASE", plus dwell, "D".

To use Cam Modulo mode set the F=128 bit or issue F=128. The position, "@P" and "RP", and the external encoder signal counter "CTR" to resets to zero when the external encoder input reaches the value of "BASE" plus dwell "D".

In cam modulo the "dwell" or idle period has been added. The dwell is defined by the command D=<expression>. When the motor reaches the last cam table value as the external encoder counter "CTR" value changes to BASE plus D, the motor position will remain at the last cam table entry for the number of external encoder counts as defined by "D". Dwell can be changed while the cam table is running. First send the new dwell value and issue "CI" which initializes the new dwell value the next time the "zero" point, end of base plus dwell or aw[0], is passed. Dwell can be set to zero.

In addition variable "CX" contains the real-time array index of the cam table. The cam table index "CX" can be used for applications where the cam tables need to be updated dynamically.

Program Example 1: Cam Modulo. Refer to Figure 9.1 .

```

MF4      'Set external encoder inputs and reset external encoder input counter.
F=128    'Set to cam modulo mode.
D=2000   'Set dwell, the number of counts.
BASE=8000 'Set number of external encoder input counts as one cam table cycle.
SIZE=8    'Set size=max array index of the cam table, where aw[array index]
aw[0]=0   'defines cam table.
aw[1]=250
aw[2]=500
aw[3]=250
aw[4]=0
aw[5]=500
aw[6]=1000
aw[7]=750
aw[8]=500
O=0      'Set current shaft position as zero point.
MC       'Set to mode cam.
G        'Start mode cam.
END
  
```

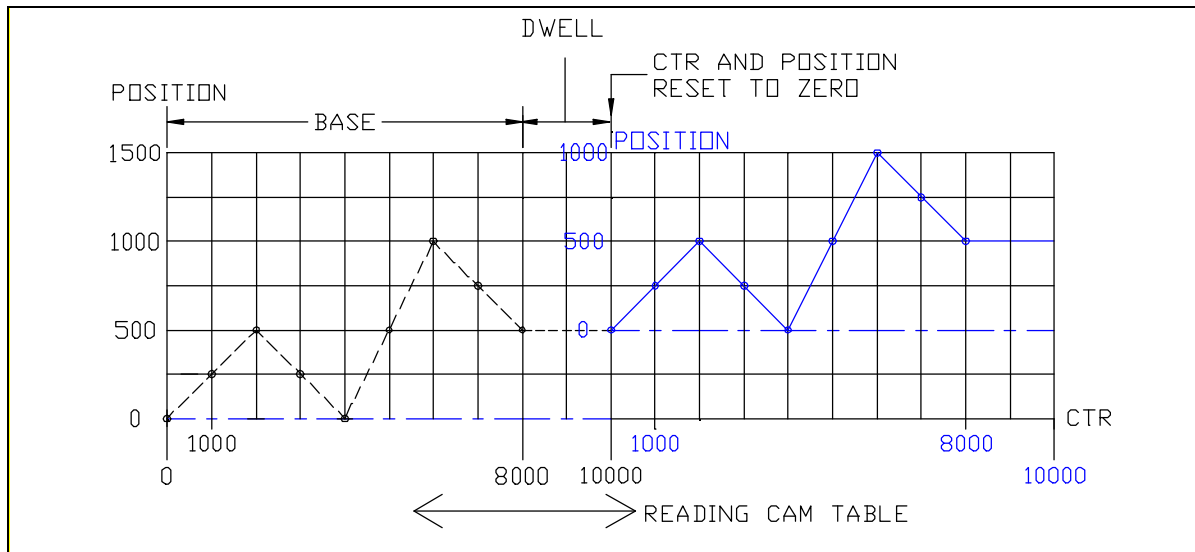


Figure 9.1: Cam Modulo Mode Example
(Two cycles shown)

9.3 Relative Positioning in Cam Mode

Cam when the external encoder input reaches BASE+D (BASE plus dwell) the operation of following the cam table will begin again relative to the (relative) position contained in the last cam table entry. The last shaft position at the end of the cam table is set as the start position of the next cam cycle.

Note: The external encoder input counter CTR and position RP, @P is not reset at the end of the cam table. Thus position may continually increase in magnitude as the external encoder counter increases, if the last cam table entry is not zero. Reversing the direction of the external encoder will reverse the process.

Value D (relative distance in Position Mode) becomes an idle or dwell distance for the cam table. For example, after running through the cam table values as the external counter value changes from 0 to BASE, the motor position will remain at the last cam table entry value for the number of external encoder counts specified in D. Upon reaching BASE+D, the operation of following the cam table will begin again, relative to the (relative) position contained in the last cam table entry.

To dynamically adjust the dwell, enter a new buffered value in D. When desiring to use the new value, issue C1. The next time the external encoder value is such that the motor crosses the zero-point boundary (between 0 and BASE plus dwell) in either direction, the buffered value in D will become the new dwell value. If the zero-point boundary is never crossed, the dwell will never change. This mechanism prevents inadvertently causing abrupt moves by jumping into the middle of the cam table when changing dwell, possibly concurrently with the external encoder changing.

Note: The position may continually increase in magnitude as the external encoder counter increases, if the last cam table entry is not zero. Reversing the direction of the external encoder will reverse the process.

Variable **CX** will contain the current (real time) index into the cam table as controlled by the external encoder value. The purpose of providing the cam table index is to simplify the process of modifying the cam table dynamically.

The cam table SIZE had a slightly different meaning for cam table in absolute position (F=0) cam mode. SIZE in relative position (F=16) cam mode is the value of the highest index of the table, that begins with index 0. Thus there are SIZE+1 entries in the cam table, aw[0] through aw[SIZE].

To use relative cam mode issue the command F=16. The position, "@P" and "RP" resets to zero when the external encoder input reaches the value of "BASE" plus dwell "D". Dwell can be set to zero.

Program Example 2: Relative Position Mode in Cam Mode (See Figure 9.2)

```

MF4      'Set external encoder input and zero external encoder count.
F=16    'Set to relative position mode in cam mode.
D=2000  'Set dwell in units of encoder counts.
BASE=8000 'Set number of external encoder input counts as one cam table cycle.
SIZE=8   'Set size=max array index of the cam table, where aw[array index].
aw[0]=0  'Define cam table.
aw[1]=250
aw[2]=500
aw[3]=250
aw[4]=0
aw[5]=500
aw[6]=1000
aw[7]=750
aw[8]=500
O=0     'Set current shaft position as zero point.
MC      'Set to mode cam.
G       'Start mode cam.
    
```

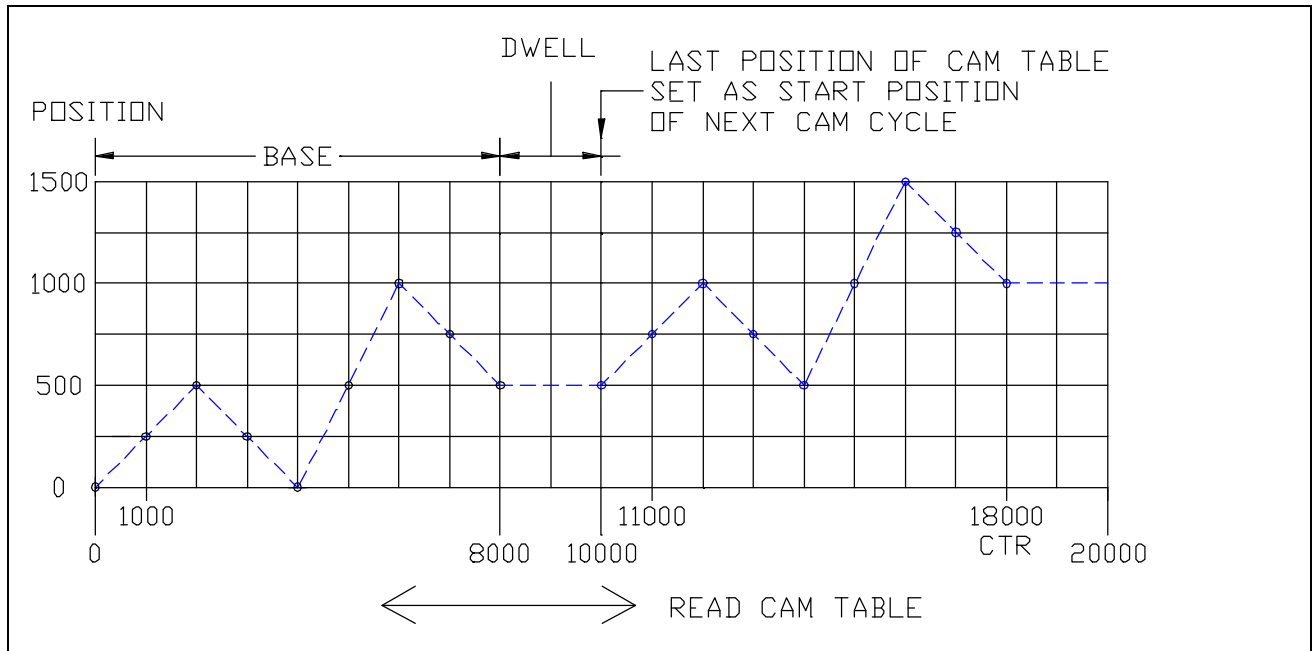


Figure 9.2: Relative Cam Mode Example
(Two cycles shown)

9.4 Mode Cam Limitations

F=16 should be issued before initializing cam mode. Otherwise absolute positioning would take immediate effect. For example, if the motor was at position 3000, and the first cam table position was 1, the motor would immediately attempt to move to 1, whereas if F=16, the motor would immediately attempt to move to 3001.

Note:

If dwell = 0, there is no external encoder value that will result in position aw[SIZE] exactly, unless aw[0] is 0. In effect, at zero dwell, aw[SIZE] would overlap with aw[0]. In the zero dwell case, if aw[0] is zero, the commanded position is the new cam origin, which would be aw[SIZE] (if starting from 0) plus aw[0] (which is zero) resulting in aw[SIZE].

EXAMPLE 3

MF4	'Set external encoder input and zero external encoder counter.
BASE=20000	' Will go less than ¼ BASE in a Sample.
SIZE=5	' SIZE is at least 4.
aw[0] 0 700 1100 2200 3300 2200.	' SIZE + 1 entries.
D=9878	
F=16	' Must be in cam relative position mode at cam initialization.
MC2	' Set to mode cam, encoder input pulses are multiplied by 2.
G	
D=9802	' buffer new dwell.
CI	' use new dwell.

9.5 On-the-Fly Parameter Changes in Cam Mode

In addition to dwell, the other cam parameters, **BASE**, **SIZE**, multiplier **MC2**, **MC4**, **MC8** may be changed on-the-fly during the execution of the cam table. Send the new values of BASE and/or SIZE to the motor followed by the **MCn (MC, MC2, MC4 or MC8)** and **CI** command to begin using the new values at the next zero-point crossing.

The issuance of any of the commands **D= dwell 0**, **BASE**, **SIZE**, **MCn**, or changing **F=16** or **128**, after the **CI** command, but before the next zero point crossing occurs, will invalidate the CI command, and no partial initialization will take place. A new **MCn** and **CI** command must be issued.

Note , F must be bit-wise equal to 16 or 128 for dwell to be operative.

If BASE and/or SIZE is being changed on-the-fly, it is necessary to issue an MCn command following the new BASE and/or SIZE assignment (preceeding the CI command, of course). This is the same as when mode cam is being started by a G MCn command, it must be issued following setting BASE and SIZE. The MCn command causes the buffered cam interval length BASE/SIZE to be calculated.

The effect of changing values **F=16** and **F=128** will not take effect in cam mode, even though the change has been made in **F**, until the **CI** command is issued and there is a zero-point crossing. Changing **F** values on-the-fly requires careful consideration of the effect, and may not result in what is imagined.

EXAMPLE 4

```
F=144          ' F=16+128, relative positioning (F=16), avoid wrap (F=128)
BASE=2000     ' quick cutoff of rough end as belt first begins moving
D=100        ' almost immediately afterward begin a normally cammed cutoff
MC2          ' double-wide first cutoff to clear cutter extension area
G
BASE=10000   ' normal cutoff profile
D=107525    ' following first normal cutoff allow normal length between cuts
MC2         ' recalculate base/size, set to mode cam. Encoder input pulses multiplied by 2
CI         ' Initialize cam parameters and which takes effect at first zero-point crossing
UAI        ' input pin A button pressing makes longer
UBI        ' input pin B button pressing makes shorter
WHILE 1
  IF UAI
    D=D+1    ' make longer
    MC2     ' recalculate base/size
    CI      ' at next zero point crossing (end dwell, begin cut)
  ENDIF
  IF UBI
    D=D-1    ' make shorter
    MC2     ' recalculate base/size
    CI      ' at next zero point crossing (end dwell, begin cut)
  ENDIF
  WAIT=100  ' max length change of 40 per second of button pressing
LOOP
END
```

EXAMPLE 5

```
b=10000      ' initial BASE
F=144
BASE=b
SIZE=10
D=0
MC2          ' required to be issued after setting BASE and SIZE to desired value
BASE=1      ' will have no effect until next MCn command
G
WHILE 1
  IF UAI
    b=b+10
  ENDIF     ' operator increases BASE
  IF UBI
    b=b-10
  ENDIF     ' operator decreases BASE
  BASE=b
  MC2      ' required to be issued after setting BASE or SIZE
  CI      ' change (if any) to take effect at next zero-point crossing
  WAIT=4000 ' doesn't change too fast if button held
LOOP
END
```

10.0 HOST MODE

10.1 Automatic Strobing of Host Mode Time

The purpose of this feature is to increase effective communication bandwidth by reducing the number of time values sent.

During host mode operation, if consecutive positions are sent to the Servida Motor (omit sending the intervening time value associated with the previous position), the Servida Motor will calculate the time value automatically, by using the previous time delta. Having sent the first two times, time values may be sent or omitted at will.

Example, constant speed:
Binary values sent to motor:

<u>Position</u>	<u><CR></u>	<u>Clock</u>	<u><CR></u>	<u>Comments</u>
0xFA 00 00 00 00	13	0xFB 00 00 00 00	13	
0xFA 00 00 10 00	13	0xFB 00 00 01 00	13	time delta 0x01 00 = 256
0xFA 00 00 20 00	13			time is 0x02 00
0xFA 00 00 30 00	13			time is 0x03 00
0xFA 00 00 34 00	13	0xFB 00 00 03 40	13	reduce time delta to 0x40 = 64
0xFA 00 00 38 00	13			time is 0x03 80
0xFA 00 00 3C 00	13			time is 0x03 C0
0xFA 00 00 40 00	13			time is 0x04 00

10.2 Orderly Host Mode Exit

The purpose of this feature is to:

- simplify and speed exit from host mode to another mode while maintaining position control;
- allow exit from host mode at a non-zero velocity while maintaining position and velocity control.

Prior to this change, users had two methods of exiting host mode:

- stop sending data points to host mode, resulting in host mode data underflow and entering mode E, losing control of the motor;
- send a long time delta to the motor while holding position, estimate when the motor would be executing the long time delta, send a G command, and likely a P=(last host mode position), to execute the next buffered mode.

During host mode operation, consecutive identical clock values may be sent to the Servida Motor to produce a zero time delta. When the host mode clock has reached the identical clock value:

- the host mode trajectory position is latched;
 - the host mode trajectory velocity is latched;
 - a "G-on-the-fly" is executed internally, using buffered mode, acceleration, velocity, position, and relative distance.
-

The position associated with the duplicated clock value is ignored.

Any buffered mode (other than host mode) initiated by the G command will be initiated:

```
MV
MP   P=expression (absolute)
MP   D=expression (relative)
MFR
MFn : where n is 4,2 or 1
MSR
MS
```

Example, servo in place at final host mode trajectory position:

```
MP
A=2
V=10000
D=0
MD
```

<Binary host mode position-time data sent to motor>

G (issued automatically if using Servida SVIEngine or SMServer primitives)
<Binary host mode position-time data ending with consecutive identical clock values (zero time delta)>

Result:

If host mode position-time data ended with the motor at zero velocity (the two positions before the duplicated clock value ignored position were identical), motor will servo at the final host mode position.

If host mode position-time data ended with the motor having a velocity (the two positions before the duplicated clock value ignored position were different), motor will:

- decelerate to a stop at acceleration A=2;
- then return to the final host mode position in a trapezoidal trajectory governed by A=2, V=10000; and finally servo in place at the final host mode position.

Example: Constant Speed

```
O=0      'begin at origin 0
MV
V=4*65536 'V of 4 counts per Sample
A=100
MD      'buffer host mode
```

Binary values sent to motor to load host mode position-time buffer:

```
<Position>      <CR> <Clock>      <CR>
0xFA 00 00 00 00 13 0xFB 00 00 00 00 13
0xFA 00 00 04 00 13 0xFB 00 00 01 00 13 ' 4*256 position cnts over 256 samples (V=4*65536)
                                0xFB 00 00 01 00 13 ' strobe position, zero time delta
```

```
G                                ' begin host mode
<pause>
RMODE      returns V
RV         returns 262144
                                ' 4*65536
```

In this example, host mode transitioned to velocity mode, maintaining constant velocity each Sample.