SMARTDRIVE
MANUAL ADDENDUM B

New Features for SmartDrives Introduced with Firmware Version v6.10

This addendum contains documentation for new IDeal™ commands and various other new features that are not included in SmartDrive manuals, or other manual addenda, due to their recent release. **Please save this documentation for future reference.**

New Features:

- **RS232 Serial Communications Test**
- **Keypad Copy-To-From**
- **New B Series Servo Motor Support**
- **Linear Servo Motor Support**
- **Position Maintenance**
- **User Definable Acceleration Maximum**
- **User Definable Output States**
  - **On Power-Up**
  - **On Fault**
  - **On Stop/Kill**
- **User Selectable EOT Switch Polarity**
- **Clear Terminal Input Buffer**
- **User Selectable Homing Mode**
- **Keypad Display Formatting**
- **User Definable Keypad Passwords**
- **Arithmetic Operators**
  !=, ++, +=n, --, -=n, >>n, <<n
- **Logical Operations on Expressions**
  &&, ||
- **New Built-In Variables**
  (AROWREL)
  (CPOS1), (CPOS2)
  (CUR1), (CUR2)
  (EPOS1), (EPOS2)
  (VEL1), (VEL2)

New IDeal™ and Serial Commands:

• BR	Break
• LU	Loop Until Condition True
• LW	Loop While Condition True
• ST#	Stop on Command
• AM	Acceleration Max
• DF	Display Format
• ET	EOT Switch Polarity
• HM	Homing Mode
• IR	In-Range/Position Maintenance DeadBand
• OE	Output States on Event
• PG	Position Maintenance Gain
• PV	Position Maintenance Max Velocity
• PW	Password
• CB	Clear Command Buffer
• PAC	Tell Commanded Position
• PAE	Tell Encoder Position

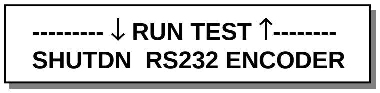
P/N PCW-5039
Revised: 09/98

Table of Contents

Table of Contents (Continued)

New RUN Menu Option

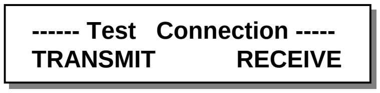
The RUN > TEST > RS232 feature has now been implemented which allows for testing and debugging of daisy chain terminal communications through the keypad thus eliminating the need for a PC terminal connection.



```
--------- ↓ RUN TEST ↑--------
SHUTDN  RS232 ENCODER
```

Testing RS232 Serial Communications

RUN > TEST > **RS232**

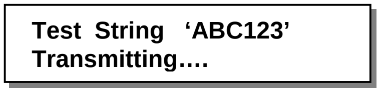


```
------ Test   Connection -----
TRANSMIT          RECEIVE
```

This feature allows for testing of the terminal serial communications port through the keypad.

Testing Serial Transmission:

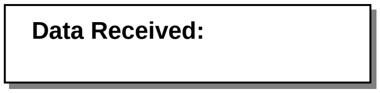
1. From the Test Connection menu press the F1 key to select the TRANSMIT option
2. The SmartDrive will now transmit the string "ABC123" every 5 seconds.



```
Test  String   'ABC123'
Transmitting….
```

Testing Serial Receive:

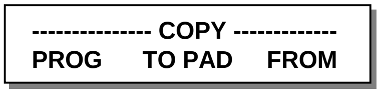
1. From the Test Connection menu press the F3 key to select the RECEIVE option
2. Any character received on the terminal port will be displayed on the keypad.



```
Data Received:
```

New COPY Menu Option

The COPY TO-FROM feature has been implemented which allows user setup and programs to be downloaded to and from the keypad. A special keypad cable (PCS-5004) provides a +5V power supply and a 9 pin D style connector for communications with Application Developer™. COPY TO-FROM requires SmartDrive version v6.00 or higher and keypad firmware v2.60 or higher. Contact IDC for firmware upgrades and cable information.



```
--------------- COPY -------------
PROG      TO PAD     FROM
```

Copy To Keypad:

1. Press F2 key to select TO PAD (**Note:** Unit address is not saved in keypad)

Copy From Pad:

1. Press F3 key to select FROM (**Note:** Unit address is not set from keypad transfer)

In order to use COPY TO-FROM with Application Developer™, connect the keypad to the PC using cable PCS-5004 and select "Retrieve All" from the Application Developer™ *Communications* menu to load data from the keypad and select "Send All" from the Application Developer™ *Communications* menu to load data to the keypad. See the Application Developer™ section in the S696X, B896X and 96X manuals for more information on using Application Developer™.

New System Configurations

New Motor Configuration

```
------- MOTOR  TYPE ------
STEPER R-SRVO L-SRVO
```

The structure of the motor type selection menu has changed in order to accommodate the addition of the built-in linear servo motor files plus the expansion of the B servo motor series. The MOTOR TYPE menu option now has three sub-menus: STEPER (for step motor drive parameters), R-SRVO (for rotary servo motors) and L-SRVO (for linear servo motors).

Configuring Stepper Motor Type

EDIT > SETUP > MOTOR > TYPE > **STEPER**

```
-- Axis One  Motor Type --
←↓      STEPPER       ↑→
```

Default:	STEPPER	(S696X)
	N/A	(B896X)
	INDEXER	(96X)

This menu option is fixed to type STEPPER in S696X SmartDrives.

Configuring R-SRVO Motor Type [MT]

EDIT > SETUP > MOTOR > TYPE > **R-SRVO**

```
-- Axis One R-Srvo Type -
←↓    B32: 110VAC    ↑→
```

Default:	N/A	(S696X)
	NONE	(B896X)
	INDEXER	(96X)

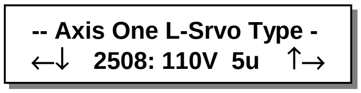
This option specifies the type of rotary servo motor connected to the B896X SmartDrive and the operating voltage level. The motor parameters used for drive configuration have been specifically tailored for IDC supplied rotary servo motors.

1. Use the ← and → keys to select an axis
2. Use the ↑ and ↓ keys to scroll through the list of IDC rotary motors
3. Press the ESC key to select

Consult the factory if you intend to use a non-IDC rotary servo motor. See Chapter 7 of the B896X Brushless Servo SmartDrive manual **RS-232 Operation** for more information on configuring the SmartDrive for a non-IDC rotary servo motor.

Configuring L-SRVO Motor Type [MT]

EDIT > SETUP > MOTOR > TYPE > **L-SRVO**



-- Axis One L-Srvo Type -
←↓ 2508: 110V 5u ↑→

Default:	N/A	(S696X)
	NONE	(B896X)
	INDEXER	(96X)

This option specifies the type of linear servo motor connected to the SmartDrive, the operating voltage level and the linear encoder resolution in microns. The motor parameters used for drive configuration have been specifically tailored for IDC supplied linear servo motors. Drive resolution, encoder resolution and gear ratio become fixed parameters and distance units are restricted to linear quantities when a L-SRVO motor type is selected. The default distance units for linear servo motors is mm, the default velocity units are mm/s and default acceleration units are seconds. Linear motors also have built-in protection against accelerations exceeding 5g regardless of acceleration units or the AMAX parameter (See the **New Mechanics Configuration:** **Configuring Acceleration Maximum [AM]** of this addendum for more information on the AMAX parameter).

1. Use the ← and → keys to select an axis
2. Use the ↑ and ↓ keys to scroll through the list of IDC linear motors. Pressing the F1 key will jump 8 items up the list and pressing the F3 key will jump 8 items down the list.
3. Press the ESC key to select

Consult the factory if you intend to use a non-IDC linear servo motor. See Chapter 7 of the B896X Brushless Servo SmartDrive manual **RS-232 Operation** for more information on configuring the SmartDrive for a non-IDC linear servo motor.

NOTE: The CL/CT force control features are currently not compatible with linear servo motors.

New Encoder Configuration

```
-- ↓ ENCODER  SETUP ↑--
MODE   E-RES   FOL-ERR
```

A new encoder mode called CLOSED LOOP PM (Position Maintenance) has been added to the selection list. Position maintenance provides post move closed loop “maintenance” of the last commanded position. This feature gives stepper systems and servo systems with an encoder on the load position correction ability. It is not recommended or necessary to use position maintenance with B896X servo drive controls since the position loop is already closed.

Configuring Encoder Mode [EM]

```
---Axis One   Enc Mode --
←↓CLOSED LOOP PM↑→
```

EDIT > SETUP > ENC > **MODE**

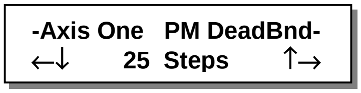
Default: OPEN LOOP (S696X, 96X)
SERVO CLOSED (B896X)

This option sets the encoder mode for each axis.

1. Use ← and → keys to select an axis
2. Use the ↑ and ↓ keys to scroll through the list of encoder modes and press ESC to select.

Encoder Mode	Description
OPEN LOOP	The OPEN LOOP position will be displayed on the keypad.
OPEN-STALL	The OPEN LOOP position will be displayed on the keypad but the encoder will be used for stall detection.
CLOSED LOOP	The encoder position is displayed on the keypad. All moves are based on encoder position and stall detection is enabled.
SERVO-CLOSED	The encoder position is displayed on the keypad. All moves are based on the commanded OPEN LOOP position and stall detection is enabled.
CLOSED LOOP PM	The encoder position is displayed on the keypad. All moves are based on encoder position, however, post move correction algorithms will keep the encoder position equal to the last commanded OPEN LOOP position. Following error is still active while in CLOSED LOOP PM mode. A following error will occur when the number of correction steps exceeds the following error value. This allows the SmartDrive to signal a fault when the displacement can not be corrected (i.e. the motor is stalled at an obstruction). Position maintenance will not attempt to correct position while navigating menus with the keypad.

Configuring Position Maintenance Deadband [IR]



```
-Axis One   PM DeadBnd-
←↓        25  Steps    ↑→
```

EDIT > SETUP > ENC > IN-RANGE > **WINDOW**

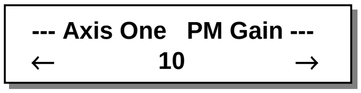
Default: 25 motor steps
Range: 0 - 99999

Position maintenance deadband is a user definable region surrounding the commanded position in which the motor shaft can reside and not be considered “out of position”. A displacement position exceeding the last commanded position +/- the deadband value, will cause position maintenance to attempt to correct the position.

1. Use ← and → keys to select an axis
2. Use numeric keys to enter a new deadband value and press ENTER then ESC to register.

NOTE: EDIT > SETUP > ENC > INRANGE > WINDOW and the IR serial command have an alternate functionality with servo SmartDrive systems. See the B8961/2 manual for more details.

Configuring Position Maintenance Gain [PG]



```
--- Axis One   PM Gain ---
←             10              →
```

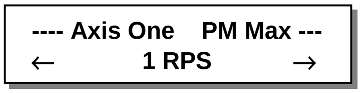
EDIT > SETUP > ENC > **PMGAIN**

Default: 10
Range: 1 - 32767

The position maintenance gain value is an integer factor used in determining the velocity at which the position maintenance correction move will travel. Correction velocity is calculated as (displacement * correction gain) in units of steps/sec. Therefore, the larger the displacement, the faster position maintenance will attempt to correct position. For example, if the correction gain is set to 3 and an active displacement of 3200 steps occurs, the correction velocity will be (3 * 3200) = 9600 steps/sec.

1. Use ← and → keys to select an axis
2. Use numeric keys to enter a new position maintenance gain and press ENTER then ESC to register.

Configuring Position Maintenance Max Velocity [PV]



```
---- Axis One    PM Max ---
←            1 RPS            →
```

EDIT > SETUP > ENC > **PMMAX**

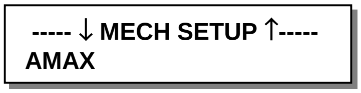
Default: 1 RPS
Range: 0.005 - 9999999

The position maintenance maximum velocity value sets a velocity limit in which position maintenance will attempt to correct position. Regardless of the magnitude of displacement or correction gain, the correction velocity will never exceed the maximum velocity setting.

1. Use ← and → keys to select an axis
2. Use numeric keys to enter a new position maintenance max velocity in the same units selected in the SETUP > MECH > VEL menu and press ENTER then ESC to register.

New Mechanics Configuration

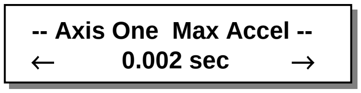
A new mechanics option AMAX (Acceleration Maximum) has been added for setting acceleration limits.



```
----- ↓ MECH SETUP ↑-----
AMAX
```

Configuring Acceleration Maximum [AM]

EDIT > SETUP > MECH > **AMAX**



```
-- Axis One  Max Accel --
←        0.002 sec       →
```

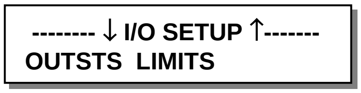
Default: 0.002 {acceleration units}
Range: 0.002 - 99999999 {acceleration units}

Acceleration maximum sets a maximum acceleration and deceleration limit for programmed move profiles in the current acceleration units. Programmed accelerations and decelerations for moves will be limited by this parameter (analogous to VMAX for velocity). Regardless of acceleration units, the absolute maximum acceleration is 0.002 seconds.

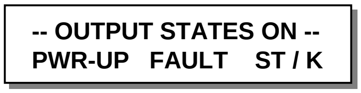
1. Use ← and → keys to select an axis.
2. Use numeric keys to enter a new acceleration maximum in the same units selected in the SETUP > MECH > ACCEL menu and press ENTER then ESC to register.

New Inputs & Outputs (I/O) Configuration

Two new I/O setup options, OUTSTS and LIMITS, have been added. OUTSTS (Output States on Event) allows user configuration of output states after power-up, fault or a Stop/Kill. LIMITS allows user configuration of the EOT (End of Travel) switch polarity. Also a new configurable input **CLR CMD BUFFER** “c” has been added.



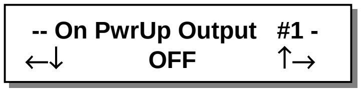
```
-------- ↓ I/O SETUP ↑-------
OUTSTS  LIMITS
```



```
-- OUTPUT STATES ON --
PWR-UP   FAULT    ST / K
```

Configuring Output States on Power Up [OEP]

EDIT > SETUP > I/O > OUTSTS > **PWR-UP**



```
-- On PwrUp Output   #1 -
←↓          OFF          ↑→
```

Default: OFF

This option sets the desired states of the outputs on power up.

1. Use ← and → keys to scroll through outputs #1- #8 and any OPTO positions configured as outputs.
2. Use the ↑ and ↓ keys to set the output state as OFF or ON and press ESC to register.

Configuring Output States on Fault [OEF]

```
--- On  Fault Output  #1 ---
←↓      NO CHANGE      ↑→
```

EDIT > SETUP > I/O > OUTSTS > **FAULT**

Default: NO CHANGE

This option sets the desired states of the outputs on a fault.

1. Use ← and → keys to scroll through outputs #1- #8 and any OPTO positions configured as outputs.
2. Use the ↑ and ↓ keys to set the output state as OFF, ON or NO CHANGE and press ESC to register.

Configuring Output States on Stop / Kill [OES]

```
-- On  ST / K  Output  #1 -
←↓      NO CHANGE      ↑→
```

EDIT > SETUP > I/O > OUTSTS > **ST / K**

Default: NO CHANGE

This option sets the desired states of the outputs on a Stop or Kill.

1. Use ← and → keys to scroll through outputs #1- #8 and any OPTO positions configured as outputs.
2. Use the ↑ and ↓ keys to set the output state as OFF, ON or NO CHANGE and press ESC to register.

Configuring End of Travel Switch Polarity [ET]

```
---- Axis One EOT Pol ---
←↓    NORM CLOSED  ↑→
```

EDIT > SETUP > I/O > **LIMITS**

Default: NORM CLOSED

This option allows configuration of the EOT switch polarity as NORM OPEN or NORM CLOSED to accommodate the use of either type of switch.

1. Use ← and → keys to select an axis.
2. Use the ↑ and ↓ keys to select NORM OPEN or NORM CLOSED and press ESC to register.

Configuring Input Definition [ID]

```
IN1: CLR CMD BUFFER
cUUUUUUU-------- ←↓↑→
```

EDIT > SETUP > I/O > > **INPUTS**

Default: UUUUUUUU

New Input Character Description

c **Clear Command Buffer**
Clears the terminal input buffer and buffered command buffer

New HOME Parameters

```
----- ↓ HOME SETUP ↑----
MODE    EDGE   SWITCH
```

The home setup parameters have been expanded to include a homing mode selection option which determines how the GH command functions.

Configuring Homing Mode [HM]

```
-- Axis One Home Mode -
←↓  SWITCH ONLY    ↑→
```

EDIT > SETUP > HOME > **MODE**

Default: SWITCH ONLY (Any OPEN LOOP encoder mode)
SWITCH THEN Z (Any CLOSED LOOP encoder mode)

The homing mode parameter establishes how a Go Home (GH) command will execute homing routines.

1. Use ← and → keys to select an axis
2. Use the ↑ and ↓ keys to scroll through the list of homing modes and press ESC to select.

Homing Mode	Description
SWITCH ONLY	GH will only home to the appropriate edge of the home switch regardless of encoder mode. This is the only mode available without an encoder.
SWITCH THEN Z	GH will home to the switch, align to the edge and then slowly move until an encoder Z pulse is found. This mode requires an encoder.
Z CHANNEL ONLY	GH will slowly move until an encoder Z pulse is found. The state of the home switch is ignored. The magnitude of the GH velocity parameter is ignored. The sign of the GH velocity parameter determines the low speed direction. This mode requires an encoder.

New Miscellaneous Setup Parameters

The Display Format (DISP) feature has now been implemented as well as the addition of user definable keypad password protection (PASWRD).

```
------ ↓MISC SETUP ↑------
DISP   STOP-RATE  TEST
```

```
------ ↓MISC SETUP ↑------
FAULT ENABLE PASWRD
```

Configuring Display Format [DF]

EDIT > SETUP > MISC > **DISP**

```
 < Quad #1 >     Quad #2
   Quad #3       Quad #4
```

```
------ Quad #1 Display -----
↓           INPUTS          ↑
```

Default: Quad #1: POS1
Quad #2: POS2 (dual axis units)
BLANK (single axis units)
Quad #3: INPUTS
Quad #4: OUTPUTS

Display format allows the user to customize the data displayed on the keypad run time screen. The run time screen has been divided into 4, 10 character configurable quadrants. The DISP menu displays labels for the 4 quadrants with carets (< >) denoting the selected quadrant.

1. Use ← ,→, ↑ and ↓ keys to move quadrant selection delimiters (< >).
2. Press ENTER to edit quadrant.

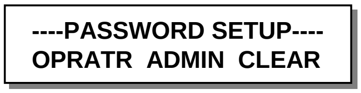
Once a quadrant is selected, there are 16 possible data types that can be displayed in that quadrant.

Data Type	Quadrant Display
BLANK	No display
POS1	Axis #1 position
POS2	Axis #2 position
POS1+UNIT	Axis #1 position with axis units
POS2+UNIT	Axis #2 position with axis units
VEL1	Axis #1 commanded velocity
VEL2	Axis #2 commanded velocity
CUR1	Axis #1 current in Amps (B896X only)
CUR2	Axis #2 current in Amps (B896X only)
INPUTS	Discreet input status (0 off, 1 on)
OUTPUTS	Discreet output status (0 off, 1 on)
OPTOS	OPTO input and output status (0 off, 1 on) as configured
SA_STATUS1	Displays SA serial command response for axis #1
SA_STATUS2	Displays SA serial command response for axis #2
SS_STATUS	Displays SS serial command response
TEXT	Display user defined text in a quadrant

3. Use the ↑ and ↓ key to scroll through the data types and press ESC to register all data types except TEXT (see below).
4. In order to define a text field, scroll to the TEXT data type and then press the ALPHA key or a number key. A cursor will appear allowing up to 10 characters to be entered. Type the desired text, press the ENTER key and then press ESC to register.

Configuring Keypad Passwords [PW]

EDIT > SETUP > MISC > **PASWRD**



----PASSWORD SETUP----
OPRATR ADMIN CLEAR

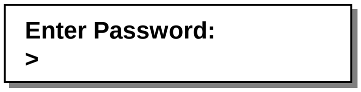
Default: None

In addition to the keypad dip switches, user definable passwords allow restricted access to the RUN, EDIT, COPY and DEL menus.

1. From the PASWORD SETUP menu, press F1 to enter an OPRATR password or press F2 to enter an ADMIN password.
2. Enter password using a maximum of 4 alpha-numeric characters only (0-9, A-Z and a-z only). See S696X, B896X and 96X manuals on how to enter alpha-numeric data on the keypad.
3. Press ENTER to register the password and ESC to exit.

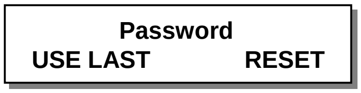
Defined Passwords	Menu Accessibility
OPRATR only	RUN, EDIT, COPY, DEL
ADMIN only	RUN, EDIT, COPY, DEL
OPRATR with ADMIN	RUN only (All RUN functions except TEST)
ADMIN with OPRATR	RUN, EDIT, COPY, DEL

If no passwords are defined, there are no menu restrictions. If passwords are defined, pressing RUN, EDIT, COPY or DEL will display the password prompt. Entering the wrong password or pressing ESC at the password prompt will return the keypad to the standard run-time display. Selecting EDIT > SETUP > MISC > PASWRD > CLEAR will delete all passwords.



Enter Password:
>

NOTE: Once a valid password is entered, the password prompt is replaced by the USE LAST, RESET prompt. Select USE LAST (F1) to use the last entered password or RESET (F3) to require the password to be reentered (i.e. require the next user to enter the password). This allows for subsequent use of the RUN, EDIT, COPY and DEL keys without requiring the password to be entered each time.



Password
USE LAST RESET

New and Updated IDeal$^{\text{TM}}$ Program Commands

BR

Break.. syntax - BR

The Break command instantly “breaks” a loop block in which it is defined and continues program execution from the loop’s terminating EB command. This allows for more complex loop conditioning than LU or LW commands.

Example:

```
(A)=0                          {Define variable A}
(B)=0                          {Define variable B}
LP                             {Define loop block}
  IF(A)>10                     {Check if A is greater than 10}
    IF2,0                      {Check if input #2 is off}
      BR                       {Break loop}
    EB
  EB
(A)++                          {Increment variable A by 1}
EB                             {BR command jumps here}
MS1,”A is greater than 10”     {Display message}
```

LU

Loop Until Condition True…………………… syntax - LUn (See Below)

The LU command defines a loop block in which loop iterations are based on a conditional result. The syntax for LU**,** which is identical to the IF command, is as follows:

Syntax(s): LUn,xx…
LUxx…(assumes first input is input 1)
LU(Mathematical expression)

Range: n = starting input number, 1-16 (SmartDrive)
x = 0, (Input Off)
x = 1 (Input On)
x ≠ 0 or 1 (Input level ignored)
Mathematical expression = Any valid conditional or logical expression

The LU loop will continue to iterate until the specified conditional is true. LU checks the conditional at the *end* of the loop block, therefore regardless if the conditional is true on the first iteration, the block is executed at least once. See the LW (Loop While Conditional True) to define loops where the conditional is checked at the *beginning* of the loop block. LU follows the same constraint of 16 nested blocks as the IF and LP commands. A GT command within a LU loop will terminate the loop, clear the loop stack and jump to the new program.

NOTE: An End of Block (EB) command must be used with every LU command.

Example #1: This loop is executed *11* times with a final position of 110 distance units.

```
(A)=0                        {Define variable A}
LU(A)=10                     {Loop until A = 10}
  DI10 GO                    {Move 10 distance units}
  (A)++                      {Increment variable A}
EB                           {End loop}
```

Example #2: This loop is executed once since the (A)<20 condition is immediately true.

```
(A)=10                       {Define variable A}
LU(A)<20                     {Loop until A < 20}
  DI10 GO                    {Move 10 distance units}
EB                           {End loop}
```

Example #3: This loop will continue to execute as long as inputs #3 and #5 are off.

```
LUXX1X1                      {Loop until inputs #3 and #5 are on}
  MS1,"Inputs 3 & 5 are off" {Display keypad message}
EB                           {End loop}
```

Example #4: This loop will continue to execute as long as input #4 is off.

```
LU4,1                        {Loop until input #4 is on}
  MS1,"Input 4 is off"       {Display keypad message}
EB                           {End loop}
```

LW

Loop While Condition True………………… syntax - LWn (See Below)

The LW command defines a loop block in which loop iterations are based on a conditional result. The syntax for LW**,** which is identical to the IF command, is as follows:

Syntax(s): LWn,xx…
LWxx…(assumes first input is input 1)
LW(Mathematical expression)

Range: n = starting input number, 1-16 (SmartDrive)
x = 0, (Input Off)
x = 1 (Input On)
x ≠ 0 or 1 (Input level ignored)
Mathematical expression = Any valid conditional or logical expression

The LW loop will continue to iterate while the specified conditional is true. LW checks the conditional at the *beginning* of the loop block, therefore if the conditional is false on the first iteration, the block is immediately skipped. See the LU (Loop Until Conditional True) to define loops where the conditional is checked at the *end* of the loop block. LW follows the same constraint of 16 nested blocks as the IF and LP commands. A GT command within a LW loop will terminate the loop, clear the loop stack and jump to the new program.

NOTE: An End of Block (EB) command must be used with every LW command.

Example #1: This loop is executed 11 times with a final position of 110 distance units.

```
(A)=0                    {Define variable A}
LW(A)<=10                {Loop while A is less than or equal to 10}
  DI10 GO                {Move 10 distance units}
  (A)=(A)+1              {Increment variable A}
EB                       {End loop}
```

Example #2: This loop is immediately skipped since the (A)>20 condition is false.

```
(A)=10                   {Define variable A}
LW(A)>20                 {Loop while A is greater than 20}
  DI10 GO                {Move 10 distance units}
EB                       {End loop}
```

Example #3: This loop will continue to execute as long as inputs #3 and #5 are on.

```
LWXX1X1                  {Loop while inputs #3 and #5 are on}
  MS1,"Inputs 3 & 5 are on"  {Display message}
EB                       {End loop}
GT[Inputs Off]           {Jump to program}
```

Example #4: This loop will continue to execute as long as input #4 is on.

```
LW4,1                    {Loop while input #4 is on}
  MS1,"Input 4 is on"    {Display message}
EB                       {End loop}
GT[Input Off]            {Jump to program}
```

ST

Stop on Input or Command……………… syntax – STn,n or ST#n,#n

Units: N/A
Range: 0-16 (Inputs)
1-2 (Axes)

ST#n,#n Syntax:

ST#1 command stops move execution on axis #1. ST,#2 stops move execution on axis #2. ST#1,#2 stops move execution on both axes.

ST#n functions identically to the STn command without the use of an input allowing program command conditional motion termination. The motor is stopped at the deceleration rate specified in the Stop Decel Rate setup parameter.

Example: Move to absolute position of 6 distance units. If (A) > 10, motion is stopped.

```
AC1,1 VE25,25 DA6,6 GI    {Move to 6 absolute distance units and GO Immediate}
IF(A)>10                  {Check value of A (Assume A was previously defined)}
  ST#1 TD1 ST,#2          {Stop motion on axis #1 wait 1 sec then stop axis #2}
EB
ST#1,#2                   {Stop motion on both axes}
```

New Arithmetic Operations

New Operators

Operator	Description
!=	Not Equal
++	Single Increment
+=n	Value Increment by n
--	Single Decrement
-=n	Value Decrement by n
>>**n**	Shift Right by n
<<**n**	Shift Left by n

Incrementing and Decrementing Variables (++, +=, --, -=)

There are four new syntaxes now supported by variables: ++ (Single Increment), += (Value Increment), -- (Single Decrement) and -= (Value Decrement). These operators will initialize any uninitialized variable to zero before incrementing or decrementing for the first time.

(Variable Name)++	Increments variable by 1
(Variable Name)+=n	Increments variable by n
(Variable Name)--	Decrements variable by 1
(Variable Name)-=n	Decrements variable by n

Logical Operations on Expressions (&&,||)

Conditional commands (IF, WT, LU, LW) now support logical operations of AND (&&) and OR (||). Two expressions may be logically AND'd or OR'd within one conditional statement. For example:

```
(A)=5 (B)=2.5               {Define variables A and B}
IF(A)>2&&(B)=2.5            {Logically AND both statements to create single conditional}
  MS1, "True"
EB
```

In the above example, the message "True" would be displayed on the keypad since BOTH statements are true, thus making the entire IF conditional true.

AND		
Expression A	**Expression B**	**A & B**
False	False	False
False	True	False
True	False	False
True	True	True

OR		
Expression A	**Expression B**	**A || B**
False	False	False
False	True	True
True	False	True
True	True	True

New Built-In Variables

Variable Name	Description	Type
(AROWREL)	Current status of any on the 4 arrow keys (See explanation below)	Read-only
(CPOS1), (CPOS2)	Commanded position of axis 1-2	Read-only
(CUR1), (CUR2)	Motor current in Amps (B896X only)	Read-only
(EPOS1), (EPOS2)	Encoder position of axis 1-2	Read-only
(VEL1), (VEL2)	Commanded velocity of axis 1-2	Read-only

Using the Built-In Variable (AROWREL)

(AROWREL) is a built-in Boolean read only variable which determines the status of any of the 4 arrow keys. When used in conjunction with (FKEY), the user can detect whether or not an arrow key is being held down. (AROWREL) will return one of the following values:

(AROWREL) = 0	One of the arrow keys is being held down.
(AROWREL) = 1	The arrow key has been released.

(AROWREL) will return key status for the 4 arrow keys only. If any other key is pressed, (AROWREL) will return zero regardless if the key is held down or not. The following is an example jog application using (AROWREL) and (FKEY):

```
[MAIN]                {Program #1}
FK12,13               {Wait for a Left or Right arrow key}
GT(FKEY)              {Jump to arrow key program #12 or #13}

[LEFTARROW]           {Program #12}
MC+                   {Enable MC mode}
AC.1                  {Start MC move}
VE1                   {Move in positive direction}
GO
LP
  IF(AROWREL)=1       {Check status of arrow key}
    VE0               {Stop MC move on key release}
    GO
    GT1               {Return to main program}
  EB
EB                    {End loop block}

[RIGHTARROW]          {Program #13}
MC+                   {Enable MC mode}
AC.1                  {Start MC move}
VE-1                  {Move in negative direction}
GO
LP
  IF(AROWREL)=1       {Check status of arrow key}
    VE0               {Stop MC move on key release}
    GO
    GT1               {Return to main program}
  EB
EB                    {End loop block}
```

New and Updated Serial Commands

New and Updated Serial Setup Commands		
Command	**Command Description**	**Syntax**
AM	**Acceleration Max** Sets the acceleration maximum (Units selected by AU command).	**<n>AMr,r**
DF	**Display Format** Configures the 4 keypad run time display quadrants. DF takes 4 parameters where i is an integer representing a display data type per quadrant. User defined text is limited to 10 characters per field. i = 0 BLANK 1 POS1 2 POS2 3 POS1 + UNIT 4 POS2 + UNIT 5 VEL1 6 VEL2 7 CUR1 8 CUR2 9 INPUTS 10 OUTPUTS 11 OPTOS 12 SA_STATUS1 13 SA_STATUS2 14 SS_STATUS “User defined text in quotes” Example: DF1,5,11,”Text Here”	**<n>DFi,i,i,i** **<n>DF”Text”,i,i,i**
EM	**Encoder Mode** Selects the encoder mode i = 0 OPEN LOOP 1 OPEN-STALL 2 CLOSED LOOP 3 SERVO-CLOSED 4 CLOSED LOOP PM	**<n>EMi,i**
ET	**End of Travel Switch Polarity** Selects the polarity of the EOT (End of Travel) switches. i = 0 NORM OPEN 1 NORM CLOSED	**<n>ETi,i**

New and Updated Serial Setup Commands

Command	Command Description	Syntax
HM	**Homing Mode** Selects homing mode i = 0 SWITCH ONLY 1 SWITCH THEN Z 2 Z CHANNEL ONLY	**<n>HMi,i**
IR	**In-Range Window (B896X only)** Sets In-Range window in motor steps	**<n>IRi,i**
IR	**Position Maintenance Deadband (S696X,96X only)** Sets position maintenance deadband in motor steps Valid as a program command using an immediate parameter only (No variables).	**<n>IRi,i**
IT	**In-Range Time (B896X only)** Sets In-Range time limit within In-Range Window in milliseconds	**<n>ITi,i**
MT	**Motor Type** Selects a built-in motor type. Note: Linear motor types are negative.	**<n>MTi,i**

i = 0 NONE		
1 B23: 110V	-1 2504: 110V 10µ	-30 3804: 220V 1µ
2 B23: 220V	-2 2504: 220V 10µ	-31 3804: 110V 0.5µ
3 B32: 110V	-3 2504: 110V 5µ	-32 3804: 220V 0.5µ
4 B32: 220V	-4 2504: 220V 5µ	-33 3806: 110V 10µ
5 B41: 110V	-5 2504: 110V 1µ	-34 3806: 220V 10µ
6 B41: 220V	-6 2504: 220V 1µ	-35 3806: 110V 5µ
7 H3: 110V	-7 2504: 110V 0.5µ	-36 3806: 220V 5µ
8 H4: 110V	-8 2504: 220V 0.5µ	-37 3806: 110V 1µ
9 OTHER	-9 2506: 110V 10µ	-38 3806: 220V 1µ
10 B12: 110V	-10 2506: 220V 10µ	-39 3806: 110V 0.5µ
11 B22: 110V	-11 2506: 110V 5µ	-40 3806: 220V 0.5µ
12 B22: 220V	-12 2506: 220V 5µ	-41 3808: 110V 10µ
13 B23H: 110V	-13 2506: 110V 1µ	-42 3808: 220V 10µ
14 B23H: 220V	-14 2506: 220V 1µ	-43 3808: 110V 5µ
15 B31: 110V	-15 2506: 110V 0.5µ	-44 3808: 220V 5µ
16 B31: 220V	-16 2506: 220V 0.5µ	-45 3808: 110V 1µ
17 B33: 110V	-17 2508: 110V 10µ	-46 3808: 220V 1µ
18 B33: 220V	-18 2508: 220V 10µ	-47 3808: 110V 0.5µ
19 B40: 110V	-19 2508: 110V 5µ	-48 3808: 220V 0.5µ
20 B40: 220V	-20 2508: 220V 5µ	-49 3810: 110V 10µ
21 B42: 110V	-21 2508: 110V 1µ	-50 3810: 220V 10µ
22 B42: 110V	-22 2508: 220V 1µ	-51 3810: 110V 5µ
23 B42: 220V	-23 2508: 110V 0.5µ	-52 3810: 220V 5µ
24 BN21: 110V	-24 2508: 220V 0.5µ	-53 3810: 110V 1µ
25 BN23: 110V	-25 3804: 110V 10µ	-54 3810: 220V 1µ
26 BN31: 110V	-26 3804: 220V 10µ	-55 3810: 110V 0.5µ
27 BN32: 110V	-27 3804: 110V 5µ	-56 3810: 220V 0.5µ
	-28 3804: 220V 5µ	-57 OTHER
	-29 3804: 110V 1µ	

New and Updated Serial Setup Commands

Command	Command Description	Syntax
OE	**Output States on Event** Configures output states on an event specified by a. OPTO positions 9-16 are only definable if configured as an output using OP command. a = P (Power-Up) F (Fault) S (Stop / Kill) i = 0 Off 1 On X No Change Example: OEF,0001XX01XXXXXXXX	**<n>OEa,iiiiiiiiiiiiiiii**
PG	**Position Maintenance Gain** Sets position maintenance correction gain. i = 1 to 32,767	**<n>PGi,i**
PV	**Position Maintenance Max Velocity** Sets position maintenance maximum correction velocity (Units specified by VU command).	**<n>PVi,i**
PW	**Password** Specifies the OPRATR and ADMIN keypad menu restriction passwords. The syntax for the PW command is PWOPRATR,ADMIN. Passwords can be a maximum of 4 characters consisting of numerical digits 0-9 or upper and lower case alpha characters A-Z. Specifying a dash (-) will clear the respective password and a null parameter will leave the respective password unchanged. Passwords can not be retrieved using the UA command. The following are examples using PW: Example #1: PW4FT,Q12h This would set the OPRATR password to 4FT and the ADMIN password to Q12h**.** Example #2: PW,New This would change the ADMIN password to New and leave the OPRATR password unchanged. Example #3: PW-,- This would clear both the OPRATR and ADMIN passwords.	**<n>PWaaaa,aaaa**

Updated Serial Supervisory Commands		
Command	**Command Description**	**Syntax**
SW	**Software/Firmware Version** **SW** returns firmware version **SW1** returns DSP, FPGA and firmware versions	**<n>SW** **<n>SW1**

New and Updated Serial Immediate Status Commands		
Command	**Command Description**	**Syntax**
CB	**Clear Command Buffer** Clears the terminal input buffer and buffered command buffer	**<n>CB**
PA	**Tell Absolute Position** Reports current position in user units based on encoder mode selected. Can report specifically commanded or encoder position when PAa,n is used. a = C (Commanded Position) E (Encoder Position) n = 1-2 axis Examples: PA2 Returns axis #2 encoder mode position PAC,1 Returns axis #1 commanded position	**<n>PAn** **<n>PAa,n**