

LOGOSOL AC/DC PROGRAMMABLE SERVO & LOGIC CONTROLLER LS-151

SOFTWARE REFERENCE

USER LEVEL COMMAND SET		2
FED	FINDS THE EDGE OF DIGITAL INPUT	2
HOM	HOMES THE AXIS	2
IN	DISPLAYS THE STATE OF DIGITAL INPUT	3
JGF	JOGS THE AXIS IN FORWARD DIRECTION	4
JGR	JOGS THE AXIS IN REVERSE DIRECTION	4
MVA	MOVES THE AXIS TO ABSOLUTE POSITION	5
MVR	MOVES THE AXIS TO RELATIVE POSITION	5
NOP	DISABLES THE CONTROLLER MAIN POWER	6
NOS	BREAKS THE SERVO LOOP	6
OUT	CONTROLS DIGITAL OUTPUTS	7
POW	ENABLES THE CONTROLLER MAIN POWER	8
ALL	DISPLAYS INFORMATION ABOUT THE CONTROLLER	8
RSP	RESTORES THE PARAMETERS	9
SAC	SETS ACCELERATION FOR THE AXIS	9
SER	ACTIVATES THE SERVO LOOP	10
SIL	SETS PID COEFFICIENT IL	10
SKD	SETS PID COEFFICIENT KD	11
SKI	SETS PID COEFFICIENT KI	11
SKP	SETS PID COEFFICIENT KP	12
STP	STOPS THE AXIS MOTION	12
SVL	SETS VELOCITY FOR THE AXIS	13
SVP	SAVES THE AXIS PARAMETERS	13
LIST OF FILES ON THE FLASH DISK		14
SYSTEM CONFIGURATION FILE		15
SYSTEM MACRO FILE IMPLEMENTATION		17

User level command set

FED	FINDS THE EDGE OF DIGITAL INPUT
Syntax:	FED <i>Switch</i> <i>Switch</i> input to be used as a home or limit switch
Description:	Finds the edge of home or limit switch. This command is essential part of the homing procedure, but it can also be useful in any generic procedure dealing with digital inputs as a source of information for motion coordination.
Example:	<pre>>FED IN6 ></pre>
HOM	FINDS THE HOME POSITION OF THE AXIS
Syntax:	HOM [<i>Switch</i>] <i>Switch</i> input to be used as a home switch (optional)
Description:	Finds the home position of the axis or finds the index of its encoder. When used without parameters, this command finds the first occurrence of the encoder index of the axis. If evoked with a parameter, the last is interpreted as a name of an input used for the homing procedure. The homing procedure provides accurate initialization of the origin (zero position) of the axis, which is mandatory for every servo-control system. The home command should be executed prior any motion or series of motions to absolute coordinates.
Example:	<pre>>HOM IN6 ></pre>

IN

DISPLAYS THE STATE OF DIGITAL INPUT

Syntax:

IN [*InputName*]

InputName name of the input to be displayed (optional)

Description:

Reads the state of digital input and displays it.

When used without parameter, this command displays the state of all digital inputs. The names and respective current states of the inputs are displayed.

If a parameter is specified, it is interpreted as an input name and the state of that input is displayed.

Note:

LS-151 has seven built-in general-purpose digital inputs and one emergency stop input. They are labeled **IN0** to **IN6** and **EMG**.

Example:

```
>IN IN5  
1  
>
```

JGF **JOGS THE AXIS IN FORWARD DIRECTION**

Syntax: **JGF** [*Velocity*]

Velocity velocity to be set for the axis (optional)

Description: Jogs the axis in forward direction.

If a parameter is specified, it is interpreted as a velocity to be set for the axis. The current velocity value is not preserved.

To stop the motion use **STP** command.

Example:

```
>JGF  
>
```

JGR **JOGS THE AXIS IN REVERSE DIRECTION**

Syntax: **JGR** [*Velocity*]

Velocity velocity to be set for the axis (optional)

Description: Jogs the axis in reverse direction.

If a parameter is specified, it is interpreted as a velocity to be set for the axis. The current velocity value is not preserved.

To stop the motion use **STP** command.

Example:

```
>JGR  
>
```

MVA **MOVES THE AXIS TO ABSOLUTE POSITION**

Syntax: **MVA** *Position*

Position absolute target position

Description: Moves the axis to the absolute target position.

This command is blocking, i.e. no other motion command can be issued before motion completion.

Example:

```
>MVA 2000  
>
```

MVR **MOVES THE AXIS TO RELATIVE POSITION**

Syntax: **MVR** *RelPosition*

RelPosition offset from the current position

Description: Moves the axis to a position, relatively specified (via *RelPosition*) with respect to the current position.

This command is blocking, i.e. no other motion command can be issued before motion completion.

Example:

```
>MVR 1500  
>
```

NOP **DISABLES THE CONTROLLER MAIN POWER**

Syntax: **NOP**

Description: Disables the controller main power.

This command turns-off the power of the servo board. All outputs are switched to their inactive state. The high voltage power relay is opened to disconnect the controller from the external power supply.

The power to the encoders and axis position is retained after **NOP** command.

Example: This command turns-off the servo amplifiers as well.

```
>NOP  
>
```

NOS **BREAKS THE SERVO LOOP**

Syntax: **NOS**

Description: Breaks the servo loop.

This command turns-off the servo loop of the axis. Current axis position remains valid because the encoder power is not turned-off.

If a brake is associated with the axis, its power will be turned-off as well in order to hold the axis in its current position.

Example: >NOS

```
>
```

OUT

CONTROLS DIGITAL OUTPUTS

Syntax:

OUT [[*OutputName*] = *NewValue*]

OutputName output name (optional)

NewValue new output state – 0 or 1 (optional)

Description:

Controls digital outputs.

When used without parameters, this command displays the state of all digital outputs. The names and respective current states of the outputs are displayed.

When only *OutputName* is specified (no *NewValue* is assigned), the current state of the output is displayed.

If both parameters are specified, the state of the digital output referenced by the first parameter is set to the value of the second parameter.

It is possible to specify more than one output/state pair in single command.

Note:

LS-151 has 5 built-in general-purpose digital outputs. They are labeled as **OUT0** to **OUT4**. The servo controller also has a relay-output controlling the main power supply. It is labeled as **MAIN**.

Examples:

```
>OUT OUT3
1
>OUT OUT3 = 0
>OUT OUT1 = 0 OUT2 = 0
>
```

POW **ENABLES THE CONTROLLER MAIN POWER**

Syntax: **POW**

Description: Enables the controller main power.

This command turns-on the power relay of the motor and starts a diagnostic sequence. All outputs are set to their initial state. The motion processor is initialized and the amplifier is enabled.

The diagnostic procedure requires the axis position to not change for 200 ms. Otherwise the power shuts down

Example:

```
>POW  
>
```

ALL **DISPLAYS INFORMATION ABOUT THE CONTROLLER**

Syntax: **ALL**

Description: Displays detailed information about LS-151 controller, containing PID filter parameters, velocity, acceleration and digital input/output states.

Example:

```
>ALL  
KP = 80  
KD = 1000  
KI = 100  
IL = 100  
VEL = 20000  
ACC = 10000  
POS = 158000  
ERR = -12  
IN0  IN1  IN2  IN3  IN4  IN5  IN6  EMG  
0    1    0    0    0    0    0    0  
OUT0 OUT1 OUT2 OUT3 OUT4 MAIN  
0    0    0    1    0    1  
The system servo is ON
```

RSP **RESTORES THE PARAMETERS**

Syntax: **RSP**

Description: Restores the parameters previously saved with **SVP** command.

The LS-151 servo controller saves the axis parameters into non-volatile memory.

Note: This command is intended for use in conjunction with **SVP** command.

Example:

```
>RSP
OK
```

SAC **SETS ACCELERATION FOR THE AXIS**

Syntax: **SAC** *Acceleration*

Acceleration new maximum acceleration to be set for the axis

Description: Sets maximum acceleration for the axis.

The specified acceleration should be nonzero. If it is too high, then either the maximum position error will be exceeded or the amplifier current protection will shut down the power of the motor.

Example:

```
>SAC 1000
>
```

SER	ACTIVATES THE SERVO LOOP
Syntax:	SER
Description:	<p>Closes the servo loop and releases the brake, if any.</p> <p>This command turns-on the power of the motor amplifier and closes the servo loop. If a brake is associated with the axis it is released automatically.</p> <p>After closing the servo loop the axis will feel stiff when trying to move it manually.</p>
Example:	<pre>>SER ></pre>
SIL	SETS PID COEFFICIENT IL
Syntax:	<p>SIL <i>Coefficient</i></p> <p><i>Coefficient</i> new value to be set for IL coefficient</p>
Description:	<p>Sets PID coefficient IL for the axis.</p> <p>The IL (integral limit) coefficient limits the restoring force of the axis performed by the integral term KI. The integral limit defines an upper boundary, which the integral sum may grow to.</p> <p>The complete set of PID coefficients has to be in harmony to control the overall behavior of the PID regulators of the servo control processor. Choosing wrong values for the motion parameters may result in a strange behavior of the axis such as jitter, vibrations or no motion at all.</p>
Example:	<pre>>SIL 100 ></pre>

SKD **SETS PID COEFFICIENT KD**

Syntax: **SKD** *Coefficient*

Coefficient new value to be set for KD coefficient

Description: Sets PID coefficient KD for the axis.

KD provides a force that is proportional to the rate of change of the position error. Correct settings contribute to an appropriate damping resulting in less perturbation.

The complete set of PID coefficients has to be in harmony to control the overall behavior of the PID regulators of the servo control processor. Choosing wrong values for the motion parameters may result in a strange behavior of the axis such as jitter, vibrations or no motion at all.

Example:

```
>SKD 1000  
>
```

SKI **SETS PID COEFFICIENT KI**

Syntax: **SKI** *Coefficient*

Coefficient new value to be set for KI coefficient

Description: Sets PID coefficient KI for the axis.

The KI coefficient provides a restoring force that grows with the time and thus insures that the static position error becomes and remains zero.

The complete set of PID coefficients has to be in harmony to control the overall behavior of the PID regulators of the servo control processor. Choosing wrong values for the motion parameters may result in a strange behavior of the axis such as jitter, vibrations or no motion at all.

Example:

```
>SKI 100  
>
```

SKP **SETS PID COEFFICIENT KP**

Syntax: **SKP** *Coefficient*

Coefficient new value to be set for KP coefficient

Description: Sets PID coefficient KP for the axis.

The KP coefficient provides a restoring force proportional to the position error. KP is used each time the PID regulators detect a deviation between the expected target and the actual position.

The complete set of PID coefficients has to be in harmony to control the overall behavior of the PID regulators of the servo control processor. Choosing wrong values for the motion parameters may result in a strange behavior of the axis such as jitter, vibrations or no motion at all.

Example:

```
>SKP 80  
>
```

STP **STOPS THE AXIS MOTION**

Syntax: **STP**

Description: Stops the axis motion using the current acceleration setting.

Example:

```
>STP  
>
```

SVL **SETS VELOCITY FOR THE AXIS**

Syntax: **SVL** *Velocity*

Velocity new velocity to be set for the axis

Description: Sets velocity for the axis.

Example:

```
>SVL 2000  
>
```

SVP **SAVES THE AXIS PARAMETERS**

Syntax: **SVP**

Description: Saves the axis parameters.

The LS-151 servo controller saves the axis parameters into a non-volatile memory.

Note: This command is intended for use in conjunction with RSP command.

Example:

```
>SVP  
OK
```

List of files on the flash disk

The firmware of LS-151 servo & logic controller consists of the following files, stored on the build-in solid state disk.

IBMBIO.SYS IBMDOS.SYS COMMAND.COM	DOS system files
CONFIG.SYS AUTOEXEC.BAT	DOS configuration files
XTTY.SYS	Serial communications driver.
LOAD.EXE	Bootstrap loader.
SYSTEM.CFG	Configuration file for the bootstrap loader.
TRANSFER.EXE	Utility to exchange files with the host via serial line.
LS-151.INI	System configuration file.
LS-151.MAC	System macro file.
MCL.EXE	The motion control kernel.
USER.MAC	User macro file.

System configuration file

```
; Logosol programmable servo & logic controller LS-151.  
; System configuration file  
;  
; Copyright 1999 Logosol, Inc.  
; For technical support contact support@logosolinc.com
```

[BOARDS]

```
LS151          0x280
```

[INPUTS]

```
IN0           LS151   IN0  LOW  
IN1           LS151   IN1  LOW  
IN2           LS151   IN2  LOW  
IN3           LS151   IN3  LOW  
IN4           LS151   IN4  LOW  
IN5           LS151   IN5  LOW  
IN6           LS151   IN6  LOW  
EMG           LS151   EMG  HIGH
```

[OUTPUTS]

```
OUT0          LS151   OUT0 HIGH OFF  
OUT1          LS151   OUT1 HIGH OFF  
OUT2          LS151   OUT2 HIGH OFF  
OUT3          LS151   OUT3 HIGH OFF  
OUT4          LS151   OUT4 HIGH OFF  
MAIN          LS151   SPS  HIGH ON
```

[AXES]

```
Name          X        rotary  
Master        LS151   0
```

[PARAMETERS]

VEL X = 100000 ; velocity
ACC X = 100000 ; acceleration
MAX X = 1000 ; maximum position error
KP X = 80 ; KP coefficient
KD X = 1000 ; KD coefficient
KI X = 100 ; KI coefficient
IL X = 80 ; IL coefficient
DS X = 256 ; derivative sampling rate
FLIMIT X = NOLIMIT ; forward limit
RLIMIT X = NOLIMIT ; reverse limit
PRO X = 0 ; velocity profile type

; profiles:

; 0 - trapezoidal velocity
; 1 - s-curve velocity
; 2 - spline velocity

[SYSTEM]

TimeSlice 5 ; kernel time-slice [ms]
Info Text ; information gathering mode
Report 0 ; report verbosity flags
Kernel ON ; kernel command set toggle
; maximum number of:
MaxSamples 1000 ; samples
MaxArrays 5000 ; array cells
MaxVars 220 ; variable slots
MaxLines 6000 ; program lines
MaxMacros 240 ; macros defined
MaxProc 120 ; procedures defined
Upload transfer /s
Download transfer /r

System macro file implementation

```
; Logosol Programmable Servo & Logic Controller LS-151.
; Generic macro file, Rev. 1.02
; Copyright 1999 Logosol, Inc.
; For technical support contact support@logosolinc.com

CONST _findx    0x0008
CONST _fnoservo 0x0080
CONST _fstop    0x0400

; variables used in PID-filter optimizer
VAR RangeKP, RangeKD, RangeKI, RangeIL

;-----
; Finds the edge of home or limit switch.
; arguments:
;   s - input to be used as a home switch
; local vars:
;   os      the initial input state
;   sfli    saved forward limit
;   srli    saved reverse limit

PROC _fed s:nn os:on sfli:on srli:on
    sfli = FLI x
    srli = RLI x
    FLI [x] = NOLIMIT
    RLI [x] = NOLIMIT
    os = IN [s]
    lbl:
    IF os
        FOR x
    ELSE
        REV x
    ENDIF
    GO x
    WAIT (os != IN [s])
    HALT x
    WAIT (STA [x] & _fstop)
    FLI x = sfli
    RLI x = srli

RETURN
```

```

; front end.
MACRO fed s:nn
    IF (STA [x] & _fnoservo)
        PRINT "fed: no servo.", #10
        RETURN
    ENDIF
    IF (s == _NA)
        PRINT "fed: input name expected.", #10
        RETURN
    ENDIF
    SUBMIT _fed s
RETURN

;-----
; "Homes" the axis or finds the encoder index.
; arguments:
;     s - input to be used as a home switch (optional)
; local variables:
;     sfli    saved forward limit
;     srli    saved reverse limit

PROC _hom s:nn sfli:on srli:on
    STROBE x = 14
    IF (s != _NA)
        _fed s
    ENDIF
    sfli = FLI x
    srli = RLI x
    FLI x = NOLIMIT
    RLI x = NOLIMIT
    LATCH x
    FOR x
    GO x
    WAIT (STA [x] & _findex)
    HALT x
    WAIT (STA [x] & _fstop)
    ABS [x] = IND [x]
    GO x
    WAIT (STA [x] & _fstop)
    DELAY 200
    POS x = 0
    FLI x = sfli
    RLI x = srli
RETURN

```

```

; front end.
MACRO hom s:nn
    IF (STA [x] & _fnoservo)
        PRINT "hom: no servo.", #10
        RETURN
    ENDIF
    SUBMIT _hom s
RETURN

;-----
; The IN command is supported by MCL Kernel itself.
; See MCL Programming Guide for more details about
; it's implementation.

;-----
; Jogs the axis in forward direction.
; arguments:
;     v - velocity (optional)
; local variables:
;     none

MACRO jgf v:nn
    IF (STA [x] & _fnoservo)
        PRINT "jgf: no servo.", #10
        RETURN
    ENDIF
    IF (v != _NA)
        vel [x] = v
    ENDIF
    FOR x
    GO
RETURN

```

```

;-----
; Jogs the axis in reverse direction.
; arguments:
;     v - velocity (optional)
; local variables:
;     none

MACRO jgr v:nn
    IF (STA [x] & _fnoservo)
        PRINT "jgr: no servo.", #10
        RETURN
    ENDIF
    IF (v != _NA)
        vel [x] = v
    ENDIF
    REV x
    GO
RETURN

;-----
; Moves the axis to absolute position.
; arguments:
;     p - new absolute position
; local variables:
;     none

PROC _mva p:nn
    ABS x = p
    GO
    WAIT ((STA [x]) & _fstop)
    ; The point of making this procedure to wait
    ; for motion completion, is to prevent submitting
    ; of another background motion command.
RETURN

```

```

; front end.
MACRO mva p:nn
    IF (STA [x] & _fnoservo)
        PRINT "mva: no servo.", #10
        RETURN
    ENDIF
    IF (p == _NA)
        PRINT "mva: position coordinate expected.",
#10
        RETURN
    ENDIF
    SUBMIT _mva p
RETURN

```

```

;-----
; Moves the axis to relative position.
; arguments:
;   p - offset from the current position
; local variables:
;   none

```

```

PROC _mvr p:nn
    REL x = p
    GO
    WAIT ((STA [x]) & _fstop)
    ; The point of making this procedure to wait
    ; for motion completion, is to prevent submitting
    ; of another background motion command.
RETURN

```

```

; front end.
MACRO mvr p:nn
    IF (STA [x] & _fnoservo)
        PRINT "mvr: no servo.", #10
        RETURN
    ENDIF
    IF (p == _NA)
        PRINT "mvr: position coordinate expected.",
#10
        RETURN
    ENDIF
    SUBMIT _mvr p
RETURN

```

```

;-----
; The NOP, NOS, OUT and POW commands are supported by
; MCL Kernel itself.
; See MCL Programming Guide for more details about
; it's implementation.

;-----
; Displays information about the controller.
; arguments:
;   p - offset from the current position
; local variables:
;   none
MACRO all
    PRINT "KP = ", KP [x], #10
    PRINT "KD = ", KD [x], #10
    PRINT "KI = ", KI [x], #10
    PRINT "IL = ", IL [x], #10
    PRINT "VEL = ", VEL [x], #10
    PRINT "ACC = ", ACC [x], #10
    PRINT "POS = ", POS [x], #10
    PRINT "ERR = ", ERR [x], #10
    PRINT "IN0  IN1  IN2  IN3  IN4  IN5  EMG", #10
    PRINT IN IN0, "    ", IN IN1, "    "
    PRINT IN IN2, "    ", IN IN3, "    "
    PRINT IN IN4, "    ", IN IN5, "    "
    PRINT IN EMG, "    ", #10
    PRINT "OUT0 OUT1 OUT2 OUT3 OUT4 MAIN", #10
    PRINT OUT OUT0, "    ", OUT OUT1, "    "
    PRINT OUT OUT2, "    ", OUT OUT3, "    "
    PRINT OUT OUT4, "    ", OUT MAIN, "    ", #10
    PRINT "the system servo is "
    IF (STA [x] & _fnoservo)
        PRINT "OFF", #10
    ELSE
        PRINT "ON", #10
    ENDIF
RETURN

```

```

;-----
; Restores parameters saved with SVP command.
; arguments:
;     none
; local variables:
;     none
MACRO rsp
    _OPEN params.sav, r
    _READ "%ld", VEL x
    _READ "%ld", ACC x
    _READ "%ld", AJE x
    _READ "%ld", DJE x
    _READ "%ld", KP x
    _READ "%ld", KI x
    _READ "%ld", KD x
    _READ "%ld", IL x
    _READ "%ld", FLI x
    _READ "%ld", RLI x
    _CLOSE
    IF (_IO != 0)
        PRINT "rsp: an i/o error has occurred.", #10
    ENDIF
RETURN

;-----
; Sets acceleration for the axis.
; arguments:
;     a - acceleration
; local variables:
;     none

MACRO sac a:nn
    IF (a == _NA)
        PRINT "sac: acceleration value expected.", #10
        RETURN
    ENDIF
    ACC x = a
RETURN

;-----
; The SER command is supported by MCL Kernel itself.
; See MCL Programming Guide for more details about
; it's implementation.

```

```

;-----
; Sets PID coefficient IL for the axis.
; arguments:
;   c - new coefficient value
; local variables:
;   none

MACRO sil c:nn
    IF (c == _NA)
        PRINT "sil: IL value expected.", #10
        RETURN
    ENDIF
    IL x = c
RETURN

;-----
; Sets PID coefficient KD for the axis.
; arguments:
;   c - new coefficient value
; local variables:
;   none

MACRO skd c:nn
    IF (c == _NA)
        PRINT "skd: KD value expected.", #10
        RETURN
    ENDIF
    KD x = c
RETURN

;-----
; Sets PID coefficient KI for the axis.
; arguments:
;   c - new coefficient value
; local variables:
;   none

MACRO ski c:nn
    IF (c == _NA)
        PRINT "ski: KI value expected.", #10
        RETURN
    ENDIF
    KI x = c
RETURN

```

```

;-----
; Sets PID coefficient KP for the axis.
; arguments:
;   c - new coefficient value
; local variables:
;   none

MACRO skp c:nn
    IF (c == _NA)
        PRINT "skp: KP value expected.", #10
        RETURN
    ENDIF
    KP x = c
RETURN

;-----
; Stops the axis motion
; arguments:
;   none
; local variables:
;   none
; The point of making this procedure to wait
; for motion completion, is to prevent submitting
; of another background motion command.
;

MACRO stp
    IF (STA [x] & _fnoservo)
        PRINT "stp: no servo.", #10
        RETURN
    ENDIF
    ABS x = POS [x]
    GO
    STOP
    WAIT (STA[x]& _fstop)
RETURN

```

```

;-----
; Sets velocity for the axis.
; arguments:
;     v - velocity
; local variables:
;     none

MACRO svl v:nn
    IF (v == _NA)
        PRINT "sac: velocity value expected.", #10
        RETURN
    ENDIF
    VEL x = v
RETURN

;-----
; Saves the axis parameters.
; arguments:
;     none
; local variables:
;     none

MACRO svp
    _OPEN params.sav, w+
    _WRITE "%ld", VEL x
    _WRITE "%ld", ACC x
    _WRITE "%ld", AJE x
    _WRITE "%ld", DJE x
    _WRITE "%ld", KP x
    _WRITE "%ld", KI x
    _WRITE "%ld", KD x
    _WRITE "%ld", IL x
    _WRITE "%ld", FLI x
    _WRITE "%ld", RLI x
    _CLOSE
    IF (_IO != 0)
        PRINT "svp: an i/o error has occurred.", #10
    ENDIF
RETURN

```