FUZZY LOGIC SERVO CONTROLLER

PROGRAMMER'S GUIDE

TABLE OF CONTENTS

1. SCOPE

1.1 System Overview

The Fuzzy Logic Servo Controller (FLSC) by Vera Ikona is an inexpensive closed loop DC motor control module that is easy to use and tune for a wide range of inertial loads and incremental optical encoder resolutions. The module has been designed as a daughter board that can be attached to various h-bridge interface boards.

The FLSC modules are slave devices that connect to a single master via a CAN bus. Communication is half-duplex whereby a command issued by the master to an individual slave is followed by an acknowledgement message from the slave to the master. If the master initiates another command to a slave before it has responded, or while it is responding, the slave will terminate the response process of the previous command. Commands received by a slave initiate deterministic non-blocking actions within the slave firmware. The slave turnaround time from command reception to the sending of the response is dependent on the amount of processing time required to either read data from memory or to validate/write data to memory. Responses do not wait for mechanical events.

Group commands to the network are used for synchronizing stop, start, CAN bus bit rate and sleep mode for power reduction. Slaves do not respond to group commands.

FLSC modules do not initiate communication exchanges. Information about the state or stored values of a particular slave must be polled by the master with read commands. The design of a servo master must carefully consider periodic polling of slave status so that events can be managed in a timely way. An example is checking the slave to see if it has halted a servo motor due to a change of state of a limit switch signal.

The communication between master and slave network has two formats depending on whether the connection is direct via the CAN bus or via the ‘USB to CAN Gateway’ device sold by Vera Ikona. Henceforth this document will refer to these two types of master as CAN master and USB master.

1.2 Scope of this document

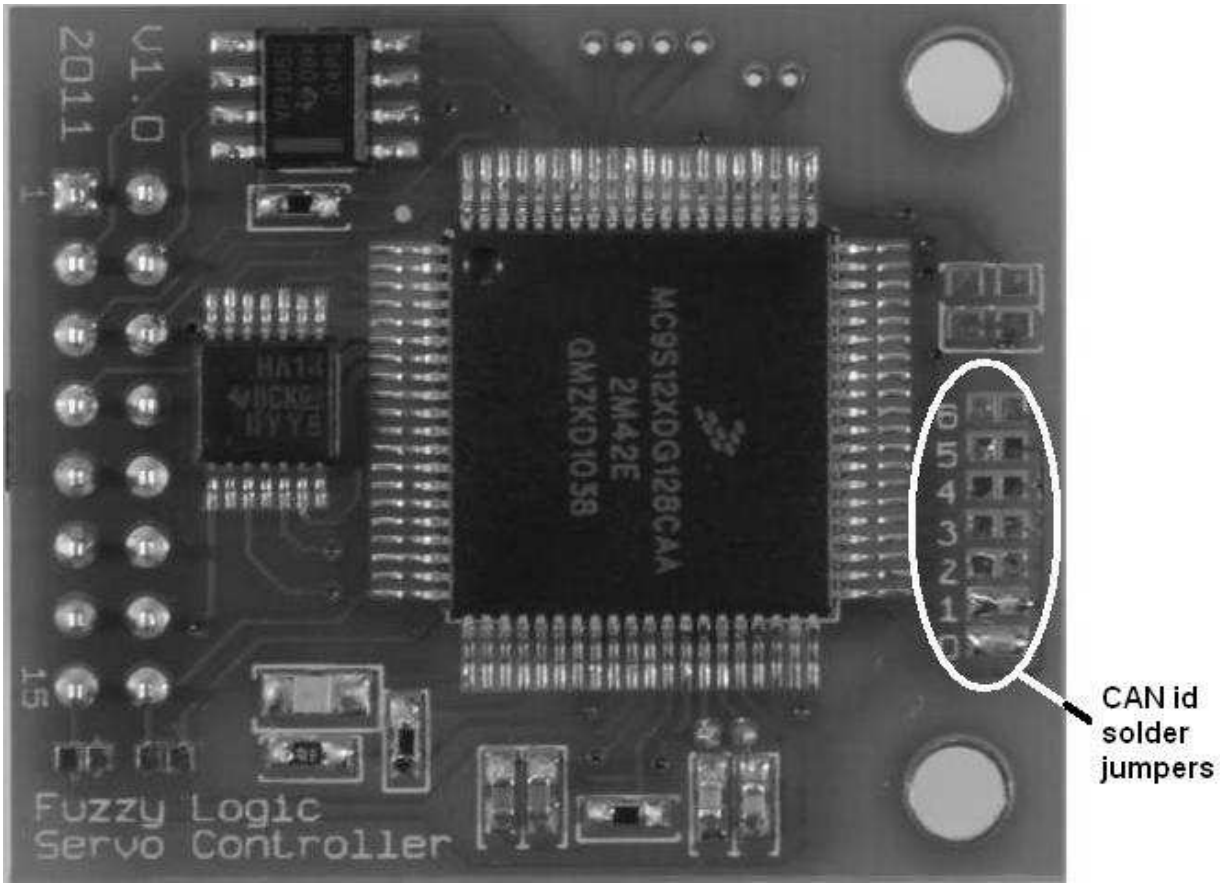
This API document explains the communication protocol and byte stream data exchanged between master control software and a slave network of FLSC modules. This document does not discuss the structure of the master software and the sequence of commands employed by a master to control a network of FLSC modules.

It is assumed that readers of this document are familiar with embedded driver development and hexadecimal notation.

2. GLOSSARY

API	Application programming interface
CAN	Controller Area Network. The CAN bus is a packet switched technology similar to ethernet (CSMA/CD). Each CAN frame contains address, data and data length information. The maximum number of bytes of data per frame (*maximum transfer unit*) is 8 bytes. Bus arbitration is automated in CAN peripherals whereby collisions are resolved by a bitwise comparison of the address fields at the start of each frame. In the event of a collision, the frame with the lower destination address is transmitted first. Bit rates up to 1 Mbit/sec are possible depending on cable length. Standard CAN addressing as per specification 2.0A is used by the fuzzy logic servo controller.
FLSC	Fuzzy Logic Servo Controller (top view on page 7)
Fuzzy logic	A logic theory that describes truth as a continuum of values. Linguistic variables allow human concepts of partial truth to be modeled.
H-bridge	An electronic circuit which enables a voltage to be applied across a load in either direction. These circuits allow DC motors to run in either direction.
Incremental optical encoder	An electro-mechanical device that converts motion into a sequence of digital pulses. Also known as a quadrature optical encoder due to the 90 degree phase relationship of the output signals which are used to determine direction of movement.

3. COMMUNICATION



3.1 Addressing

The FLSC module CAN address is determined by the shorting the numbered solder jumpers on the top side of the module. Users configuring a network of servo modules must set a unique non-zero value for each unit. Modules with no jumpers shorted (zero address) will not initialize. Modules with conflicting addresses will generate errors on the CAN bus that prevent communication. An error code will flash on the diagnostic LED of effected modules to indicate these CAN addressing issues (see section 6).

Standard identifiers are used exclusively by the FLSC CAN network. Standard identifiers are 11 bits in length which define an address range of 0x0 to 0x7FF. The base address used for CAN messages sent from the master to the FLSC network is 0x600. Master messages that use this base address are broadcast (group) messages intended for all slaves on the network. Slaves do not respond to messages with this address.

The CAN address for a specific slave module is the addition of the slave id with the base address. A module with only jumper 0 populated with solder is addressed by the master as 0x601. The largest number that can be set via the solder jumpers is 127 and therefore the slave address range is 0x601 to 0x67F. The CAN address of slave response messages to the master is the addition of 0x100 to the slave address and therefore the range of slave response addresses is correspondingly 0x701 to 0x77F. CAN bus arbitration gives priority to lower address messages and therefore any master-to-slave message on the bus will always be transmitted before a slave to master message in the event of a collision. CAN messages that are aborted due to collisions are automatically resent when the bus is available and does not require user intervention.

3.2 Message Format

All master commands and slave responses conform to a byte stream format that allows message data to be parsed and tested for integrity.

<header><len><data[0]>…<data[len – 1]><LRC>

Typically a CAN master is an embedded system with direct access to the CAN bus by an on-chip peripheral or by a CAN controller IC. This type of master requires CAN driver software that can split the message data over multiple CAN frames so that communication is perceived as a single contiguous stream of data in the upper levels of the application. CAN driver details are not presented in this document.

The software for USB masters treats messages as a single contiguous series of bytes and can ignore the work done by the gateway to split/merge the information over multiple CAN frames. Typically USB masters are devices with a standard OS and USB host capability such as a desktop computer. Messages are passed between the USB master and the gateway as serialized data via royalty-free drivers provided by FTDI (makers of the USB FIFO chip inside the gateway device). Developers of USB master software are limited to the drivers provided by FTDI at: http://www.ftdichip.com/Drivers/D2XX.htm

Header:
Master-to-slave and slave-to-master messages are delimited with the 0xAA byte. Master-to-gateway and gateway-to-master messages are delimited with the 0xCC byte.

Len:
Number of data bytes contained in the data section of the message. This number excludes the header, length and LRC bytes. Message data length cannot be greater than 255 bytes.

Data:
By convention the first byte of data for all CAN bus messages is the command number. The minimum data payload for a master message is a single byte. Slave response messages contain a minimum of three bytes of data. The second and third bytes are the

upper and lower bytes of the 16 bit return code respectively (henceforth referred to as retval.h and retval.l). This return code is in respect to the command issued only. System errors for the servo controller are obtained via status read commands (see section 4.8). Note that all multiple byte value types are transmitted in big endian order (most significant to least significant). The identifier field inherent to CAN frames makes it unnecessary for address information to be included with the data.

USB messages are formatted identically to CAN messages however the 2 byte destination address is situated at the beginning of the data. In the case of USB master-to-slave messages, the gateway device removes the 2 byte address information from the USB data, re-calculates the len and lrc bytes and forwards the message onto the CAN bus via multiple CAN frames with the proper CAN identifier. In the case of USB slave-to-master messages, the gateway device assembles the data from the slave response CAN frames into a buffer, inserts the 2 byte address, re-calculates the len and lrc bytes and forwards the message via the USB bus to the PC master.

LRC:
Longitudinal Redundancy Check – The two's complement of the least significant byte of the sum of the values of the length byte and each data byte. The sum does not include the header byte (or the lrc byte itself, obviously).

4. SERVO COMMANDS

Table 4-0-1 lists the 10 master commands for controlling FLSC modules.

Table 4-0-1. Master Commands

VALUE	COMMAND	FUNCTIONAL DESCRIPTION
0	CMD_ENUMERATE	Get FLSC id and set CAN bit rate control
1	CMD_SERVO_CTL	Get/set Servo hardware controls
2	CMD_LIMIT_CTL	Get/set position limit switch, ATD limit and IRQ controls
3	CMD_MOVE_CTL	Get/set motion controls
4	CMD_FILE_CTL	Get/set non-volatile storage of control settings
5	CMD_FUZZY_CTL	Get/set fuzzy logic controls
6	CMD_ZERO	Set zero position
7	CMD_STATUS	Get module status and clear sticky errors
8	CMD_STATS	Get servo feedback for visualization and tuning
9	CMD_SLEEP	Set network into power saving mode

The master commands and slave responses documented in this section are presented as a sequence of bytes. Byte stream sequences are from the perspective of a CAN master and do not include the additional address information passed in the USB byte stream between the gateway and the PC. CAN driver details with respect to addressing and spanning of information across multiple frames is not presented.

Some commands are complex because the slave action is conditional on the values of read/write and option bytes. The option byte is a bitfield whose bit values determine the order and type of additional optional data to be read or written. The byte stream tables of each command highlight the optional data bytes in **bold** typeface.

Turnaround times for each command are measured with the gateway timer which has a 125usec precision. Elapsed time is from when the gateway writes the master message to its CAN driver to when the response message has completely returned. Turnaround times were measured with a single slave on the network. These times are valid for networks with multiple slaves if the master is designed to wait for slave responses before sending a new command onto the bus. Since CAN arbitration gives messages of lower address priority, a slave will have to wait for the master to stop sending messages before it can put its response message onto the bus. If there are multiple slaves waiting for bus access, the one with the lowest id gets access first. The designer of the master must consider the size of the network in combination with the CAN bus bit rate and turnaround times to maximize bus loading. If the master is designed to asynchronously receive slave responses from multiple slaves it must consider the delays inherent in CAN arbitration for the slaves with a higher address. For example, if the master issues a zero command to slave node 0x601 and then immediately issues a zero command to slave node 0x602, the response from the first slave will wait for the master to complete

its command to the second slave and the response from the second slave will wait for the response from the first slave to complete.

Some master-to-slave commands can be addressed to the base address (0x600) rather than to an individual slave. These commands are characterized as having a group mode.

4.1 ENUMERATE (CMD_ENUMERATE)

The enumerate command has a slave mode and a group mode. A succession of slave mode enumeration commands are typically the first commands that the master will place on the CAN bus for network discovery and slave identification.

4.1.1 Slave mode

In slave mode this command is used to detect the presence of a specific slave on the network. This is the only command where the slave response data does not send a return code. The slave returns the 16-bit firmware version number in place of the two return code bytes that normally follow the command byte.

Table 4-1-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x01	# data bytes (excludes header, len and lrc)
cmd	0x00	CMD_ENUMERATE
lrc	0xFF	LRC of bytes between header and LRC

Table 4-1-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x00	CMD_ENUMERATE
version high	0x01 – 0xFF	Firmware version upper byte - major release #
version low	0x00 – 0xFF	Firmware version lower byte - minor release #
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-1-3. Slave Mode Turnaround

Bit Rate	Elapsed time (millisec)
125Kbps	1.625
500Kbps	.625
1Mbps	.375

4.1.2 Group mode

When this command is addressed to the base network address (group mode) it is a request for all FLSC nodes on the network to set their CAN bit rate to the specified value.

Table 4-1-4. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x02	# data bytes (excludes header, len and lrc)
cmd	0x00	Enum command #
CAN bit rate	0x00 – 0x02	Byte values are listed in table 4-1-5.
lrc	0xFE – 0xFC	LRC of bytes between header and LRC

Slave response:
None for group commands

Table 4-1-5. CAN bus bitrates

VALUE	H-BRIDGE
0x00	125Kbps (default)
0x01	500Kbps
0x02	1Mbps

The bit rate switching regime for the USB master is to issue the group command ten times before instructing the gateway to change its bit rate. CAN masters will simply change the CAN driver bit rate after issuing ten group commands. Re-enumeration of the network should be repeated to verify that the bus is capable of supporting the set bit rate.

FLSC modules that do not have the SERVO_CTL_FILE (see section 4.5) stored in its file system are initialized with the 125Kbit/sec bit rate. The USB to CAN gateway device sold by Vera Ikona lacks non-volatile storage and always starts up at the default bit rate. In the case where the FLSC network has been pre-configured to use the CAN bus at a faster rate, the gateway device must be instructed to switch to the faster rate and then clear the CAN errors for each slave caused by the initial bit rate mismatch.

Table 4-1-6 shows the maximum bit rate for various lengths of shielded twisted pair cable.

Table 4-1-6. CAN bus cabling

Bus Length (m)	Maximum Bit Rate
500	125 Kbps
100	500 Kbps
30	1 Mbps

4.2 SERVO CONTROL (CMD_SERVO_CTL)

The servo control command controls fundamental servo settings that are required for FLSC operation. When a FLSC module starts out of reset it automatically searches for the SERVO_CTL_FILE (see section 4.5) in the file system to populate the servo control variables. A FLSC module that lacks this file (ie: a new uninitialized module) requires a write of the servo control command before servo operations can be performed. The smallest group of controls that must be set by this command to allow operation are servo rate, h-bridge type and h-bridge pwm frequency.

The Servo Control command has a slave mode only. Group addressing of this command is ignored by the slave network.

4.2.1 Servo Control Write

Table 4-2-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x0A	# data bytes (excludes header, len and lrc)
cmd	0x01	CMD_SERVO_CTL
r/w	0x00	Write = 0; read = 1
op	0x00 – 0x3F	Bitfield to determine which optional controls to include. Bit values are described in table 4-2-7
Servo rate	0x00 – 0x0C	Byte values are described in table 4-2-8. Included when bit 0 of op byte set.
H-bridge type	0x00 – 0x03	Byte values are described in table 4-2-9. Included when bit 1 of op byte set.
H-bridge pwm	0x00 – 0x04	Byte values are described in table 4-2-10. Included when bit 1 of op set.
Encoder divider	0x00 – 0x04	Byte values are described in table 4-2-11. Included when bit 2 of op byte set.
Trapezoidal Kick	0x00 – 0x01	Boolean (off:on = 0:1). Included when bit 3 of op byte set.
Power at move done	0x00 – 0x01	Boolean (off:on = 0:1). Included when if bit 4 of op byte set.
GPIO control	0x00 – 0x05	Byte values are described in table 4-2-12. Included when bit 5 of op byte set.
Motor polarity	0x00 – 0x01	Boolean (reverse:forward = 0:1). Included when if bit 6 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-2-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x01	CMD_SERVO_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-2-3. Write Turnaround (all options)

Bit Rate	Elapsed time (millisec)
125Kbps	2.75
500Kbps	.875
1Mbps	.625

4.2.2 Servo Control Read

Table 4-2-4. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x01	CMD_SERVO_CTL
r/w	0x01	Write = 0; read = 1
op	0x00 – 0x3F	Bitfield to determine which optional controls to read. Bit values are described in table 4-2-7
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-2-5. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x04 – 0x0B	# data bytes (excludes header, len and lrc)
cmd	0x01	CMD_SERVO_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
op	0x00 – 0x3F	Bitfield to determine which optional controls to include. Bit values are described in table 4-2-7
Servo rate	0x00 – 0x0C	Byte values are described in table 4-2-8. Included when bit 0 of op byte set.
H-bridge type	0x00 – 0x03	Byte values are described in table 4-2-9. Included when bit 1 of op byte set.
H-bridge pwm	0x00 – 0x04	Byte values are described in table 4-2-10. Included when bit 1 of op set.
Encoder divider	0x00 – 0x04	Byte values are described in table 4-2-11. Included when bit 2 of op byte set.
Trapezoidal Kick	0x00 – 0x01	Boolean (off:on = 0:1). Included when bit 3 of op byte set.
Power at move done	0x00 – 0x01	Boolean (off:on = 0:1). Included when if bit 4 of op byte set.
Gpio control	0x00 – 0x05	Byte values are described in table 4-2-12. Included when bit 5 of op byte set.
Motor polarity	0x00 – 0x01	Boolean (reverse:forward = 0:1). Included when if bit 6 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-2-6. Read Turnaround (all options)

Bit Rate	Elapsed time (millisec)
125Kbps	2.75
500Kbps	.875
1Mbps	.5

4.2.3 Control Options and Details

Table 4-2-7. Op bitfield controls

BIT	CONTROL	# BYTES	DATA DESCRIPTION
0	Servo rate	1	Servo rates in table 4-2-8
1	H-bridge type H-bridge pwm	1 1	H-bridge types in table 4-2-9 PWM values in table 4-2-10
2	Encoder divider	1	Divider values in table 4-2-11
3	Trapezoidal move done kick	1	Boolean
4	Power on move done	1	Boolean
5	GPIO type	1	GPIO types in table 4-2-12
6	Motor polarity	1	Boolean

The op bitfield for the write version of the command describes the type of data included in the byte stream written to the FLSC module. The op bitfield for the read version of the command instructs the FLSC module which data to return in the response byte stream. The order of data sent or received is determined by the bit control (bit 0 is first). The op byte is echoed in the slave response of a read command to assist master parsing of returned optional data.

Table 4-2-8. Servo Rates

VALUE	SERVO RATE
0x00	None (default)
0x01	None
0x02	None
0x03	None
0x04	2^4 =16Hz
0x05	2^5 = 32Hz
0x06	2^6 = 64Hz
0x07	2^7 = 128Hz
0x08	2^8 = 256Hz
0x09	2^9 = 512Hz
0x0A	2^10 = 1024Hz
0X0B	2^11 = 2048Hz
0X0C	2^12 = 4096Hz

The servo rate is the number of fuzzy servo iterations executed per second. A FLSC module with a stored servo rate that is less than 4 is considered uninitialized and will not perform servo motor operations.

Table 4-2-9. H-Bridge Types

VALUE	H-BRIDGE

0x00	None (default)
0x01	LMD18200
0x02	SIMPLE-H
0x03	OSMC 3-2

H-bridge hardware selection is currently limited to three types; the National Semiconductor LMD18200 as well as the Simple-H and OSMC h-bridges sold by Robot Power. A FLSC module with a stored h-bridge type of zero is considered uninitialized and will not perform servo motor operations.

When the FLSC module is in the uninitialized state all h-bridge control lines are set as inputs. It is strongly suggested that external pull resistors be installed on the h-bridge control lines to prevent undesired motor activation by a floating logic level. Likewise caution is advised for changing the h-bridge type control while a motor is attached to the h-bridge since changes to the control lines may cause undesired motor activation.

Table 4-2-10. H-Bridge PWM

VALUE	PWM Frequency
0x00	None (default)
0x01	1005.5 Hz (1KHz)
0x02	9.8 KHz (10KHz)
0x03	15.7 KHz (16KHz)
0x04	19.6 KHz (20KHz)

The maximum pwm frequency is typically specified by the h-bridge datasheet. The 20KHz frequency is a popular choice because it is outside the normal range of human hearing. An OSMC board with all 16 MOSFETs installed has a 16KHz maximum pwm frequency. A FLSC module with a stored h-bridge pwm rate of zero is considered uninitialized and will not perform servo motor operations.

The **Encoder divider** control is used to decrease the resolution of the optical encoder by a factor of 2, 4, 8 or 16. This is a useful feature to reduce encoder feedback that overflows the signed 8-bit range of the fuzzy engine. By default this value is set to 0 (no division).

Table 4-2-11. Encoder Divider

VALUE	Encoder Divider
0x00	1 (default)
0x01	2
0x02	4
0x03	8
0x04	16

The '**Trapezoidal Kick'** byte is only relevant to trapezoidal motion. It can be selected to provide a 'kick' to the final destination position when the motion profile has completed.

The default setting is off. The kick is simply swapping trapezoidal mode with position mode when the deceleration phase of the trapezoidal profile ends.

The '**Power at move done'** byte controls whether the servo motor is kept energized after the completion of a position or trapezoidal move. The default setting is to keep the motor energized on motion completion.

The '**GPIO control**' byte determines whether the FLSC GPIO (general purpose input output) pin is used as an input or an output. If the type is an output, the state of the pin is set immediately. If an output toggle type is set then the output value will be toggled when the sleep state is entered and exited. Note that the toggle feature will not function on sleep exit if the SERVO_CTL_FILE is not present in the file system. Reads of an input configured GPIO returns the state of the pin. Reads of an output configured GPIO return the configuration setting. A write of either of the input values sets the GPIO as an input. The default setting for the GPIO pin is input.

Table 4-2-12. GPIO type

VALUE	TYPE
0x00	Input (w); pin value = 0(r)
0x01	Input (w); pin value = 1(r)
0x02	Output low
0x03	Output high
0x04	Output low; toggle on suspend
0x05	Output high; toggle on suspend

The '**Motor polarity**' byte controls the output polarity of the h-bridge. This control is used to match the direction of motor rotation with the optical encoder feedback. By convention the INC direction of pwm motion must generate an increasing encoder count. If a INC pwm motion produces a decreasing encoder count then flipping the value of this control will correct the relationship between encoder and motor rotation without having to physically reverse the motor or encoder connections.

4.3 LIMIT CONTROL (CMD_LIMIT_CTL)

The Limit Control command is used to control the sensing of and response to external events. The digital limits are used to detect absolute servo positions. The ATD is used to detect current consumption via the analog-to-digital conversion peripheral so that the servo can respond to resistance that causes the motor to stall or work overly hard. The FLSC response to a limit is immediate and requires no intervention from the master.

The Limit Control command has a slave mode only. Group addressing of this command is ignored by the slave network.

Limit A and Limit B are general purpose digital inputs which are typically connected to external switches to indicate end of travel positioning for motor rotation in each direction. Limit switches can be optical or mechanical with either logic high or logic low states to indicate a limit event. External pull resistors must be provided to prevent a false positive from floating logic on these pins.

The Z limit is available for optical incremental encoders that provide an index signal. This digital limit is used to indicate the absolute rotational position of the encoder wheel.

The IRQ limit can be used as a deadman stop. Multiple servo modules connected by their IRQ pin to a common line can be configured to immediately stop when the line is forced to a logic low level.

.

The properties of the digital limits are Boolean and are controlled with single bits inside bitfield bytes. Bitfield controls for the digital limits A, B and Z (encoder index) allow the user to enable/disable the limit, set the pin value of the limit (rising or falling edge), set the motor stop type at the limit (stop details are covered in section 4.4) and whether the servo position is zeroed at the limit. A maximum of one limit can be selected to zero the servo. Like the other digital limits, the IRQ limit is controlled by the same enable/disable and stop bitfield bytes, however It does not use the pin value or zero bitfield bytes. The IRQ pin always detects a low level state and cannot be configured to zero the servo position.

The ATD limit controls consist of an 8-bit value and a 16-bit dwell period. The value byte represents the 0 to 5 volt scale in 255 increments whereby each bit represents 19.6 millivolts. Setting the value to zero disables ATD limit detection as well as the ATD peripheral of the FLSC module. The dwell period is the minimum number of servo cycles that must consecutively execute with a detected voltage greater than or equal to the set ATD value before an ATD limit is considered true. The ATD limit will cause a soft motor stop. The ATD limit cannot be configured to automatically zero the servo position. **It is imperative that the external current sense circuitry be designed to output DC voltages strictly within the 0V to 5V range. Voltages outside of this range will damage the FLSC module.** Optional signal conditioning components on the FLSC module can be added by the user to improve filtering of voltage transients.

4.3.1 Control Options and Details

The op bitfield byte specifies which optional data is transferred for read and write commands. Table 4-3-1 describes the optional data associated with the bit controls of the op byte.

Table 4-3-1. Op byte bitfield controls

BIT	CONTROL	DESCRIPTION
0	Digital limit en	Read/write the enable bitfield for digital limits
1	Digital limit pin value	Read/write the pin value bitfield for digital limits
2	Digital limit stop	Read/write the stop bitfield for digital limits
3	Digital limit zero	Read/write the zero bitfield for digital limits
4	ATD Limit	Read/write the value and dwell period for ATD limits

The bits of each digital limit bitfield byte conform to table 4-3-2. In the case of the zero bitfield, the IRQ bit is ignored and a maximum of one bit can be set. The IRQ bit of the pin value byte is ignored.

Table 4-3-2. Digital bitfield controls

BIT	DIGITAL LIMIT
0	Limit A
1	Limit B
2	Limit Z (optical encoder index)
3	IRQ

4.3.2 Limit Control Write

If the op bitfield byte has any of bits 0-3 set to write optional digital limit controls then an additional digital AND bitfield is required. This byte is ANDed with each of the optional digital limit bitfields to confine changes to specific limits. This permits optional bitfield bits with a zero value to not unset the corresponding control for a digital limit that the user does not intend to operate on.

Table 4-3-3. Digital AND bitfield controls

BIT	CONTROL	DESCRIPTION
0	Limit A	If set then Lim A bits of en/value/stop/zero bitfields are significant
1	Limit B	If set then Lim B bits of en/value/stop/zero bitfields are significant
2	Limit Z	If set then Lim Z bits of en/value/stop/zero bitfields are significant
3	IRQ	If set then IRQ bits of en/stop bitfields are significant

Table 4-3-4. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x0B	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x00	Write = 0; read = 1
op	0x01 – 0x1F	Bitfield to determine which optional controls to include (see table 4.3.1)
digital AND	0x00 – 0x0F	AND bitfield for writes to digital limits. Included when any of bits 0-3 of op byte set.
digital limit enable	0x00 – 0x0F	Bitfield to enable/disable digital limits. Included when bit 0 of op byte set.
digital limit value	0x00 – 0x0F	Bitfield to set digital limit pin values. Included when bit 1 of op set.
digital limit stop	0x00 – 0x0F	Bitfield to set stop type for digital limit. Included when bit 2 of op byte set.
digital limit zero	0x00 – 0x0F	Bitfield to set position zero for a digital limit. Included when bit 3 of op byte set.
ATD value	0x00 – 0xFF	Voltage limit for external current sense circuitry. Included when bit 4 of op byte set.
ATD dwell	0x00 – 0xFF	Bits 15-8 of 16-bit atd dwell period. Included when bit 4 of op byte set.
ATD dwell	0x00 – 0xFF	Bits 7-0 of 16-bit atd dwell period. Included when bit 4 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-3-5. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-3-11. Turnaround Timing (write all)

Bit Rate	Elapsed time (millisec)
125Kbps	2.875
500Kbps	.875
1Mbps	.625

Limit Control Write example 1:

Limits A and B are activated for rising edge detection. Limit A detection will cause a soft stop and zero the servo position. Limit B detection will cause a hard stop. Limit Z, IRQ and ATD limits are not affected by this command.

Table 4-3-6. Write Example 1 Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x08	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x00	Write = 0; read = 1
op	0x0F	Write all digital bitfield controls – no ATD controls
digital AND	0x03	Digital write operations on LimA and LimB
digital limit enable	0x03	Enable LimA and LimB
digital limit value	0x03	Limit set on rising edge pin values for both limits
digital limit stop	0x02	Soft stop for LimA and hard stop for LimB
digital limit zero	0x01	Zero servo position on detection of LimA
lrc		LRC of bytes between header and LRC

Table 4-3-7. Write Example 1 Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
retval high	0x00	Bits 15-8 of 16-bit return value (0x0000 = SUCCESS)
retval low	0x00	Bits 7-0 of 16-bit return value
lrc		LRC of bytes between header and LRC

Limit Control Write example 2:
Disable limit B and enable limit Z without affecting edge, stop or zero information for limits B or Z. A, IRQ or ATD limits are not affected by this command. The slave response will be the same format as write example 1 if no errors occur.

Table 4-3-8. Write Example 2 Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x05	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x00	Write = 0; read = 1
op	0x01	Write enable digital bitfield control only
digital AND	0x06	Digital write operations on limit B and Z
digital limit enable	0x04	Disable limit B and enable limit Z
lrc		LRC of bytes between header and LRC

Limit Control Write example 3:
The IRQ limit is enabled for a hard stop and an ATD limit of 2.5V is set for a 100 servo cycle dwell period. A, B and Z limits are not affected by this command. The slave response will be the same format as write example 1 if no errors occur.

Table 4-3-9. Write Example 3 Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x09	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x00	Write = 0; read = 1
op	0x15	Write enable and stop digital bitfield controls + ATD
digital AND	0x08	Digital write operation on IRQ limit
digital limit enable	0x08	Enable IRQ
digital limit stop	0x08	Soft stop for IRQ
ATD value	0x80	2.5V is half of 5V range => 0x80 is half of 0xFF
ATD dwell	0x00	Upper byte of dwell period
ATD dwell	0x64	Lower byte of dwell period = 100
lrc		LRC of bytes between header and LRC

Limit Control Write example 4:

An ATD limit of 1V is set for a 1000 servo cycle dwell period. Digital limits are not affected by this command. The slave response will be the same format as write example 1 if no errors occur.

Table 4-3-10. Write Example 4 Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x06	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x00	Write = 0; read = 1
op	0x10	ATD only
ATD value	0x33	1V = 0x33 (one fifth of 0xFF range)
ATD dwell	0x03	Upper byte of dwell period = 3 * 256 = 768
ATD dwell	0xE8	Lower byte of dwell period = 232 (dwell = 768 + 232)
lrc		LRC of bytes between header and LRC

4.3.3 Limit Control Read

The read command returns the control settings that have been stored in the FLSC module via a limit control write or via the loading of a previously stored limit file from the FLSC eeprom file system. Unlike the write version of the command, reads do not use an AND byte to specify which digital limit bitfield bits are valid. In the case of a read, every bit of the bitfield is significant.

Note that the actual states and values of limit hardware are obtained via the status command (see section 4.8).

Table 4-3-12. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	Limit control command
r/w	0x01	Write = 0; read = 1
op	0x01 – 0x1F	Bitfield to determine which optional controls to read back in slave response (see table 4.3.1)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-3-13. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x0B	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
op	0x01 – 0x1F	Echo of op bitfield sent by master
digital limit enable	0x00 – 0x0F	Bitfield of enable bits for all digital limits. Included when bit 0 of op byte set.
digital limit value	0x00 – 0x0F	Bitfield of pin values for all digital limits. Included when bit 1 of op byte set.
digital limit stop	0x00 – 0x0F	Bitfield of stop bits for all digital limits. Included when bit 2 of op byte set.
digital limit zero	0x00 – 0x0F	Bitfield of zero bits for all digital limits. Included when bit 3 of op byte set.
ATD value	0x00 – 0xFF	Voltage limit for external current sense circuitry. Included when bit 4 of op byte set.
ATD dwell	0x00 – 0xFF	Bits 15-8 of optional 16-bit atd dwell period. Included when bit 4 of op byte set.
ATD dwell	0x00 – 0xFF	Bits 7-0 of optional 16-bit atd dwell period. Included when bit 4 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Limit Control Read example:
Read all control bitfields for the digital limits only. The returned values of this example indicate that:

- Limit A is enabled with a rising edge value, soft stop and no zeroing of position.
- Limit B is enabled with a rising edge value, soft stop and no zeroing of position.
- Limit Z is disabled with a falling edge value, soft stop and zeroing of position.
- IRQ is enabled with a hard stop.

Table 4-3-14. Read Example Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
r/w	0x01	Write = 0; read = 1
op	0x0F	Read all digital limit bitfield controls
lrc		LRC of bytes between header and LRC

Table 4-3-15. Read Example Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_LIMIT_CTL
retval high	0x00	Bits 15-8 of 16-bit return value (0x0000 = SUCCESS)
retval low	0x00	Bits 7-0 of 16-bit return value
op	0x0F	Read all digital limit bitfield controls
digital limit enable	0x0B	Limits A, B and IRQ enabled, limit Z disabled
digital limit value	0x03	Limits A, B use a rising edge, limit Z a falling edge
digital limit stop	0x08	IRQ has a hard stop, limits A, B and Z have soft stops
digital limit zero	0x04	Limit Z will zero the servo position.
lrc		LRC of bytes between header and LRC

Table 4-3-16. Turnaround Timing (read all)

Bit Rate	Elapsed time (millisec)
125Kbps	2.75
500Kbps	.875
1Mbps	.5

4.4 MOVE CONTROL (CMD_MOVE_CTL)

The move control command controls the start, stop, trajectory and type of motion.

Trajectory values written by the master are buffered such that they have no affect on the system until the start bit is set. Modified position, velocity and acceleration values can all be started while the servo is in operation from a previous move control command without causing interruption or discontinuity if the motion type is unchanged.

This command has a slave mode and a group mode. The group mode is limited to start and stop operations for coordination of slaves.

4.4.1 Move Control Write (Slave Mode)

Table 4-4-5. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x16	# data bytes (excludes header, len and lrc)
cmd	0x03	CMD_MOVE_CTL
r/w	0x00	Write = 0; read = 1
op	0x01 – 0x3F	Bitfield to determine which optional controls to include. Bit values are described in table 4-4-1
Stop type	0x00 – 0x01	Boolean values are described in table 4-4-2 Included when bits 0 of op byte set.
Position	0x00 – 0xFF	Position bits 31-24 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 23-16 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 15-8 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 7-0 (signed 32-bit) Included when bit 1 of op byte set.
Velocity	0x00 – 0x0F	Velocity bits 19-16 (signed 20-bit) Included when bit 2 of op byte set.
Velocity	0x00 – 0xFF	Velocity bits 15-8 (signed 20-bit) Included when bit 2 of op byte set.
Velocity	0x00 – 0xFF	Velocity bits 7-0 (signed 20-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 31-24 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 23-16 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 15-8 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 7-0 (unsigned 32-bit)

		Included when bit 2 of op byte set.
PWM direction	0x00 – 0x02	Byte values are described in table 4-4-3. Included when bit 3 of op byte set.
PWM duty cycle	0x00 – 0x64	pwm mode duty cycle Included when bit 3 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop decel bits 31-24 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop decel bits 23-16 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop decel bits 15-8 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop decel bits 7-0 (unsigned 32-bit) Included when bit 4 of op byte set.
Start motion mode	0x00 – 0x05	Byte values are described in table 4-4-4. Included when bit 5 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-4-6. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x03	CMD_MOVE_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-4-7. Turnaround Timing (write all except stop bit)

Bit Rate	Elapsed time (millisec)
125Kbps	4
500Kbps	1.25
1Mbps	.875

4.4.2 Move Control Read (Slave Mode)

Table 4-4-8. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x03	CMD_MOVE_CTL
r/w	0x01	Write = 0; read = 1
op	0x02 – 0x1F	Bitfield to determine which optional controls to read back in slave response (see table 4.4.1)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-4-9. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x15	# data bytes (excludes header, len and lrc)
cmd	0x03	CMD_MOVE_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
op	0x01 – 0x1F	Echo of op bitfield sent by master
Position	0x00 – 0xFF	Position bits 31-24 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 23-16 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 15-8 (signed 32-bit) Included when bit 1 of op byte set.
Position	0x00 – 0xFF	Position bits 7-0 (signed 32-bit) Included when bit 1 of op byte set.
Velocity	0x00 – 0x0F	Velocity bits 19-16 (signed 20-bit) Included when bit 2 of op byte set.
Velocity	0x00 – 0xFF	Velocity bits 15-8 (signed 20-bit) Included when bit 2 of op byte set.
Velocity	0x00 – 0xFF	Velocity bits 7-0 (signed 20-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 31-24 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 23-16 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 15-8 (unsigned 32-bit) Included when bit 2 of op byte set.
Acceleration	0x00 – 0xFF	Acceleration bits 7-0 (unsigned 32-bit) Included when bit 2 of op byte set.
PWM direction	0x00 – 0x02	Byte values are described in table 4-4-3. Included when bit 3 of op byte set.
PWM duty	0x00 – 0x64	pwm mode duty cycle

cycle		Included when bit 3 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop deceleration bits 31-24 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop deceleration bits 23-16 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop deceleration bits 15-8 (unsigned 32-bit) Included when bit 4 of op byte set.
Hard stop deceleration	0x00 – 0xFF	Hard stop deceleration bits 7-0 (unsigned 32-bit) Included when bit 4 of op byte set.
Current motion mode	0x00 – 0x05	Byte values are described in table 4-4-4. Included when bit 5 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-4-10. Turnaround Timing (read all)

Bit Rate	Elapsed time (millisec)
125Kbps	3.75
500Kbps	1
1Mbps	.875

4.4.3 Group mode

A group mode write of the move control command is used to synchronize the start and stop of slave modules on the network. The start bit is ignored if the stop bit is set. The stop bit will always stop all modules in motion however the start bit will only start the motion of slave modules that are not in a sticky error condition and which have a programmed motion or motion mode pending a start command. For example, if a slave has completed a trapezoidal motion and has not been updated with a new goal position then a group trapezoidal start will produce no effect for that slave.

Table 4-4-11. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x05	# data bytes (excludes header, len and lrc)
cmd	0x03	CMD_MOVE_CTL
r/w	0x00	Write = 0; read = 1
op	0x01 – 0x3F	Bitfield to determine which optional controls to include. Bit values are described in table 4-4-1
Stop type	0x00 – 0x01	Boolean values are described in table 4-4-2 Included when bits 0 of op byte set.
Start motion mode	0x00 – 0x05	Byte values are described in table 4-4-4. Included when bit 5 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Slave response:
None for group commands

4.4.4 Control Options and Details

Table 4-4-1. Op bitfield controls

BIT	CONTROL	# BYTES	DATA DESCRIPTION
0	Stop motion	1	Stop type table 4-4-2
1	Position	4	Encoder count (signed 32-bit)
2	Velocity Acceleration	3 4	Encoder count/sec (signed 20-bit) Encoder count/secsec (unsigned 32-bit)
3	PWM direction PWM duty cycle	1 1	PWM direction table 4-4-3 Unsigned char (0 – 100)
4	Hard stop deceleration	4	Encoder count/secsec (unsigned 32-bit)
5	Start motion	1	Motion type table 4-4-4

The op bitfield byte specifies which optional data is transferred for reads and writes of the move control command. Table 4-4-1 describes the optional data associated with the bit controls of the op byte. The order of data is fixed by the bit position; if bit 0 is set then the first optional data to follow the op byte is the stop motion control, etc. The op byte is echoed to the master in a read command to assist parsing of optional data. The stop and start bits are ignored by the slave in the case of a read.

Note that a read operation returns move control values previously set by a write operation. Current position, velocity and acceleration information for a servo in motion are obtained via the status command (section 4.8).

Table 4-4-2. Stop Types

VALUE	STOP
0x00	Soft
0x01	Hard

If bit 0 of the op byte is set in a write operation then the slave will stop the servo motor and ignore all other op bits. It is not a stored value and has no significance for a read operation. The stop types are soft or hard. A soft stop is simply de-energizing the motor so that is can stop via the mechanical resistance of the system. The behaviour of a hard stop depends on whether the hard stop deceleration variable has a non-zero value. If the value is zero then the hard stop is a command to servo to the current position via position mode (very hard!). If the hard stop deceleration value is non-zero then the motor will decelerate at that rate to zero velocity.

If bit 1 of the op byte is set the position variable is read from or written to the FLSC module. This value is the goal position when a start POSITION or TRAPEZOIDAL mode command is issued. Position is a signed 32-bit range (-2147483648 to -2147483647) measured in encoder counts. This range is reduced by the encoder divider setting of the servo control command.

If bit 2 of the op byte is set the velocity and acceleration variables are read from or written to the FLSC module. These variables are used to control acceleration, maximum

velocity and deceleration of the servo when the slave is commanded to start motion in TRAPEZOIDAL or VELOCITY modes. Velocity is a signed 20-bit range (-524288 to 524287) measured in encoder counts per second. Acceleration is an unsigned 32-bit range measured in encoder counts per secondsecond. The range of values for velocity and acceleration are unaffected by encoder division set in the servo control command.

Velocity and Acceleration are set simultaneously due to the interdependence of these values with respect to boundary conditions of the system. A SRV_ERR_ACC_LO error will be returned if the acceleration is so low relative to the velocity that the time required to achieve the final speed is greater than 65535 servo ticks (16 seconds at the highest servo rate of 4KHz). A SERVO_ERR_ACC_HI error will be returned if the acceleration is so high relative to the velocity that no ramping is possible. In the unlikely situation where the combined velocity and acceleration result in a ramp distance greater than the position bounds a SRV_ERR_DISP_RNG error is generated. Any of these error conditions will prevent an update of the velocity or acceleration variables.

Table 4-4-3. PWM Direction

VALUE	DIRECTION
0x00	OFF
0x01	INCREASING
0x02	DECREASING

If bit 3 of the op byte is set the pwm variables are read from or written to the FLSC module. The PWM controls are used when the slave is commanded to start motion in the PWM or PWM (POSITION LIMITED) modes. The direction of motor rotation will produce either an increasing or decreasing optical encoder count. A commanded pwm direction that results in encoder behaviour opposite to expectation will generate an error to alert the user that motor polarity or encoder polarity need to be switched. All motion modes cannot function properly if these polarities do not match. The pwm duty cycle byte is unsigned and bound by a maximum value of 100.

If bit 4 of the op byte is set the hard stop deceleration variable is read from or written to the FLSC module. The hard stop deceleration value is used to decelerate the motor to zero speed in the case of a hard stop command or in the case of a limit response that has been configured to use a hard stop. A zero value is interpreted as an extremely hard stop whereby the motor will servo to the current position regardless of current speed. This is the only move control variable that can be stored in the eeprom file system (see section 4.5). All other variables are considered volatile.

Table 4-4-4. Motion Type

VALUE	TYPE
0x00	NONE
0x01	TRAPEZOIDAL
0x02	VELOCITY
0x03	PWM
0x04	POSITION
0x05	PWM (POSITION LIMITED)

If bit 5 of the op byte is set in a write operation then the slave will start the servo motor in the motion type determined by the start motion byte. It is not a stored value and has no significance for a read operation. Note that a motion type must be stopped before a different motion type can be started. TRAPEZOIDAL mode commands the slave to servo the motor to the set target position using the velocity and acceleration settings. VELOCITY mode commands the slave to accelerate/decelerate the motor to the set velocity setting using the set acceleration. PWM mode sets the raw pwm and the direction controls for the h-bridge. POSITION mode commands the slave to servo the motor to the set target position without any regard for velocity and acceleration settings. This mode is at the core of all the non-pwm modes. PWM (POSITION LIMITED) mode is a special mode that limits pwm motion to the set position. If the direction is opposite such that the position cannot be reached then an error is returned. This command is useful for testing motor behaviour in systems pwm where mechanical constraints limit travel.

Note that a sticky error which has not been cleared by a status command write will inhibit motion start (see CMD_STATUS section).

Group use of the start bit will start motion for all slaves except in the case of sticky errors or in the case of TRAPEZOIDAL / POSITION motions where the target position of the slave is equal to its current position.

Note that in slave mode the NONE mode value will return an error, but in group mode it will cause all slaves that have been zeroed with the zero command to initiate a trapezoidal move to the zero position provided that the velocity and acceleration properties are non-zero and valid. Thus a group move with NONE motion mode can be used as a coordinated homing command for multiple slaves that have been properly initialized.

4.5 FILE CONTROL (CMD_FILE_CTL)

The FLSC file system is located in the internal eeprom of the 9S12X mcu. This command controls the storage and retrieval of slave settings encapsulated by the CMD_FUZZY_CTL, CMD_SERVO_CTL and CMD_LIMIT_CTL commands.

Wear leveling is implemented whereby deleted files are left in place so as to reduce the overuse of eeprom cells. An asynchronous data compression phase automatically begins when a partition becomes full due to repeated delete/write operations. A double partition scheme is implemented to prevent the loss of stored data during the asynchronous data compression phase if a mcu reset occurs due to power loss. Each partition is 1Kbyte in size.

The actual values of the files are not transferred on the CAN bus by this command; rather the commanded operation is internal such that current settings can be stored for very fast retrieval. A key feature of this arrangement is to allow fuzzy controls to be loaded to the fuzzy inference engine in mid-motion without disturbance to fuzzy processing so that servo behaviour can be adjusted seamlessly for variable loads.

The file command has a slave mode only. Group addressing of this command is ignored by the slave network.

4.5.1 File Control Read/Write/Delete

Table 4-5-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x0B	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_FILE_CTL
r/w/d	0x00 – 0x02	Byte indicating read/write/delete operation Byte values are described in table 4-5-4.
op	0x01 – 0x03	Bitfield to determine which file type to operate on. Bit values are described in table 4-5-5.
File number	0x00 – 0x03	File number
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-5-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x02	CMD_FILE_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-5-3. Turnaround Timing (read all)

Bit Rate	Elapsed time (millisec)
125Kbps	3.25
500Kbps	1.875
1Mbps	1.625

4.5.2 Control Options and Details

Table 4-5-4. RWD control byte

VALUE	OPERATION
0x00	Write
0x01	Read
0x02	Delete

The rwd byte determines whether the command is a write, read or delete operation. Reads are synchronous whereby the internal variables stored by a file are loaded immediately and any error is passed back to the user in the slave response. Writes and deletes are asynchronous due to the latencies of eeprom programming and thus the return value of the slave response for these commands pertains only to the success or failure to schedule the operation. A write of a file that already exists (update) automatically deletes the original file after writing the new version if there are changes to the content of the file.

Table 4-5-5. OP byte bitfield controls

BIT	FILE	DESCRIPTION
0	SERVO_CTL_FILE	Servo control settings file
1	LIMIT_CTL_FILE	Limit control settings file
2	FUZZY_IO_FILE	Fuzzy membership functions file
3	FUZZY_RULES_FILE	Fuzzy rules file
4		Delete all files (rwd = Delete)

The op byte is a bitfield that determines which file type to load, store or delete.

Due to the latencies inherent in eeprom writes, the resulting error code from a write or delete operation must be polled by the master. A zero op byte value for a read operation will return the current results of asynchronous file write or file delete operations. Errors resulting from a write or delete operation are sticky and are cleared by a write command with the op byte set to zero.

If bit 0 of the op byte is set then the 7 variables associated with the servo control command plus the CAN bit rate are stored, loaded or deleted. This file is always assigned a file number of zero so that there can only be one version of the servo control configuration.

If bit 1 of the op byte is set then the digital limit, ATD limit and hard stop deceleration variables are stored, loaded or deleted.

If bit 2 of the op byte is set then the fuzzy input membership function arrays and fuzzy output singleton array are stored, loaded or deleted.

If bit 3 of the op byte is set then the fuzzy rules array is stored, loaded or deleted.

If bit 4 of the op byte is set and the rwd byte is set for a delete then all files will be deleted and the FLSC module will be considered in an uninitialized state after the next power on reset.

The file number byte of the master byte stream sets the file number if a single file is operated on and must be in the range of 0 to 3. The file number byte is ignored in the following cases

- multiple file type operations (the file number is forced to zero)
- SERVO_CTL_FILE operation (only one servo setting is permitted)
- deletion of all files
- op byte = 0 (get/set the error code for asynchronous file operation)

On startup, the slave searches the file system for all 4 file types with file number zero and automatically writes the variables controlled by these files if found.

If the SERVO_CTL_FILE doesn't exist at startup then the slave CAN bit rate is set to 125Kbps and the slave is in an uninitialized state with respect to motion control commands until the master has issued the CMD_SERVO_CTL with the minimum data required (see section 4.2). Note that the warnings with respect to h-bridge logic lines as described in section 4.2.1 apply to a FLSC module with a stored servo control configuration. Since the SERVO_CTL_FILE is automatically loaded on power up it is possible for an erroneous setting to inadvertently activate the motor. This is a possible scenario when FLSC modules with stored servo control settings are swapped between IO boards with different h-bridge interfaces. Deletion of files prior to swapping modules is prudent.

If the "Limit control settings file" #0 is not found then no limits are set.

If either of the fuzzy control files #0 are not found then the default hardcoded default values are used for the fuzzy controls.

4.6 FUZZY CONTROL (CMD_FUZZY_CTL)

This command reads and writes the fuzzy rules array, fuzzy input membership function arrays and fuzzy output singleton array. Servo tuning is achieved through the selection of optimal array values. The tuning process is not discussed in this document.

FLSC modules that have not had their fuzzy controls initialized by a file read or prior successful writes of this command will have a default set of arrays. The default fuzzy control values of an uninitialized FLSC module are unlikely to provide good tuning for the servo but they are well formed and usable. Validation of new values prevents the user from corrupting valid fuzzy control data.

The fuzzy inference engine used by the FLSC module has a 5x5 rule base with 8-bit resolution. Fuzzification is performed by comparing the crisp position error input to the position error membership functions and the crisp position change input to the speed membership functions. Unweighted rule evaluation is applied to the fuzzified inputs to build a table of outputs that are defuzzified via a weighted average of singletons to generate a pwm output.

The fuzzy control command has a slave mode only. Group addressing of this command is ignored by the slave network.

4.6.1 Fuzzy Control Write

Table 4-6-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x7B	# data bytes (excludes header, len and lrc)
cmd	0x05	CMD_FUZZY_CTL
r/w	0x00	Write = 0; read = 1
op	0x01 – 0x03	Bitfield to determine which optional controls to include. Bit values are described in table 4-6-8
NEGLRG mem func pt1		Point 1 of NEGLRG membership function Included when bit 0 of op byte set.
NEGLRG mem func pt2		Point 2 of NEGLRG membership function Included when bit 0 of op byte set.
NEGLRG mem func sl1		Slope 1 of NEGLRG membership function Included when bit 0 of op byte set.
NEGLRG mem func sl2		Slope 2 of NEGLRG membership function Included when bit 0 of op byte set.
… (32 bytes not shown for brevity)		Points and slopes for NEGSML, OK, POSSML, POSLRG, NEGFST, NEGSLW, SAME, POSSLW Included when bit 0 of op byte set.
POSFST mem func pt1		Point 1 of POSFST membership function Included when bit 0 of op byte set.
POSFST mem func pt2		Point 2 of POSFST membership function Included when bit 0 of op byte set.
POSFST mem func sl1		Slope 1 of POSFST membership function Included when bit 0 of op byte set.
POSFST mem func sl2		Slope 2 of POSFST membership function Included when bit 0 of op byte set.
NEGHI singleton		NEGHI value of output singleton array. Included when bit 0 of op byte set.
NEGLO singleton		NEGLO value of output singleton array. Included when bit 0 of op byte set.
ZERO singleton		ZERO value of output singleton array. Included when bit 0 of op byte set.
POSLO singleton		POSLO value of output singleton array. Included when bit 0 of op byte set.
POSHI singleton		POSHI value of output singleton array. Included when bit 0 of op byte set.
rule #1 IF	0x00 – 0x04	Rule 1 ‘if position error is’ byte. Included when bit 1 of op byte set.
rule #1 AND	0x05 – 0x09	Rule 1 ‘and speed is’ byte. Included when bit 1 of op byte set.
rule #1 THEN	0x0A – 0x0E	Rule 1 ‘then output is’ byte. Included when bit 1 of op byte set.
… (69 bytes not		If/and/then bytes for rules 2 through 24

shown for brevity)		Included when bit 1 of op byte set.
rule #25 IF	0x00 – 0x04	Rule 25 'if position error is' byte. Included when bit 1 of op byte set.
rule #25 AND	0x05 – 0x09	Rule 25 'and speed is' byte. Included when bit 1 of op byte set.
rule #25 THEN	0x0A – 0x0E	Rule 25 'then output is' byte. Included when bit 1 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-6-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x05	CMD_FUZZY_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value (big endian)
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value (big endian)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

4.6.2 Fuzzy Control Read

Table 4-6-3. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x05	CMD_FUZZY_CTL
r/w	0x01	Write = 0; read = 1
op	0x01 – 0x03	Bitfield to determine which optional controls to include. Bit values are described in table 4-6-8
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-6-4. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03 – 0x7C	# data bytes (excludes header, len and lrc)
cmd	0x05	CMD_FUZZY_CTL
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
op	0x01 – 0x1F	Echo of op bitfield sent by master
NEGLRG mem func pt1		Point 1 of position membership function 1 Included when bit 0 of op byte set.
NEGLRG mem func pt2		Point 2 of position membership function 1 Included when bit 0 of op byte set.
NEGLRG mem func sl1		Slope 1 of position membership function 1 Included when bit 0 of op byte set.
NEGLRG mem func sl2		Slope 2 of position membership function 1 Included when bit 0 of op byte set.
… (32 bytes not shown for brevity)		Points and slopes for NEGSML, OK, POSSML, POSLRG, NEGFST, NEGSLW, SAME, POSSLW Included when bit 0 of op byte set.
POSFST mem func pt1		Point 1 of velocity membership function 5 Included when bit 0 of op byte set.
POSFST mem func pt2		Point 2 of velocity membership function 5 Included when bit 0 of op byte set.
POSFST mem func sl1		Slope 1 of velocity membership function 5 Included when bit 0 of op byte set.
POSFST mem func sl2		Slope 2 of velocity membership function 5 Included when bit 0 of op byte set.
NEGHI singleton		Point 1 of output singleton array. Included when bit 0 of op byte set.
NEGLO singleton		Point 2 of output singleton array. Included when bit 0 of op byte set.
ZERO singleton		Point 3 of output singleton array. Included when bit 0 of op byte set.

POSLO singleton		Point 4 of output singleton array. Included when bit 0 of op byte set.
POSHI singleton		Point 5 of output singleton array. Included when bit 0 of op byte set.
rule #1 IF	0x00 – 0x04	Rule 1 'if position error is' byte. Included when bit 1 of op byte set.
rule #1 AND	0x05 – 0x09	Rule 1 'and speed is' byte. Included when bit 1 of op byte set.
rule #1 THEN	0x0A – 0x0E	Rule 1 'then output is' byte. Included when bit 1 of op byte set.
… (69 bytes not shown for brevity)		If/and/then bytes for rules 2 through 24 Included when bit 1 of op byte set.
rule #25 IF	0x00 – 0x04	Rule 25 'if position error is' byte. Included when bit 1 of op byte set.
rule #25 AND	0x05 – 0x09	Rule 25 'and speed is' byte. Included when bit 1 of op byte set.
rule #25 THEN	0x0A – 0x0E	Rule 25 'then output is' byte. Included when bit 1 of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Verification of fuzzy variables by the FLSC module make the turnaround times longer for the write operation than for the read operation.

Table 4-6-5. Turnaround for i/o arrays (op bit 0 set, if op bit 1 clear)

Bit Rate	Write Elapsed time (msec)	Read Elapsed time (msec)
125Kbps	7.625	7
500Kbps	2.25	1.5
1Mbps	1.375	1.25

Table 4-6-6. Turnaround for rules (op bit 1 set, if op bit 0 clear)

Bit Rate	Write Elapsed time (msec)	Read Elapsed time (msec)
125Kbps	11.375	10.5
500Kbps	3.375	2.125
1Mbps	2	1.875

Table 4-6-7. Turnaround for both rules and i/o arrays (both op bits 0 & 1 set)

Bit Rate	Write Elapsed time (msec)	Read Elapsed time (msec)
125Kbps	16.75	15.25
500Kbps	4.875	2.875
1Mbps	2.875	2.75

4.6.3 Control Options and Details

The op byte is a bitfield that determines which optional controls to read or write. The order of data is fixed by the bitfield position; if bit 0 is set then the first optional data to follow the op byte are the fuzzy membership functions, etc. The op byte is echoed to the master in a read command to simplify parsing of optional data.

Table 4-6-8. Op Bitfield controls

BIT	CONTROL	# BYTES	DATA DESCRIPTION
0	Fuzzy trapezoidal membership functions	45	5 * 4 bytes for each input position function 5 * 4 bytes for each input velocity function 5 bytes for output singleton
1	Fuzzy rules	75	25 * 3 bytes for fuzzy rules

If bit 2 of the op byte is set then the fuzzy control command reads or writes the fuzzy rules array. The byte values for the fuzzy rule mnemonics must conform to the following 3 tables. A write of the rules array validates the ranges of the if/and/then byte of each rule.

Table 4-6-9. Fuzzy 'IF' rule values (input)

MNEMONIC	VALUE	DESCRIPTION
NEGLRG	0x00	large negative position error
NEGSML	0x01	small negative position error
OK	0x02	No position error
POSSML	0x03	small positive position error
POSLRG	0x04	large positive position error

Table 4-6-10. Fuzzy 'AND' rule values (input)

MNEMONIC	VALUE	DESCRIPTION
NEGFST	0x05	large negative change in position
NEGSLW	0x06	small negative change in position
SAME	0x07	No change in position
POSSLW	0x08	small positive change in position
POSFST	0x09	large positive change in position

Table 4-6-11. Fuzzy 'THEN' rule values (output)

MNEMONIC	VALUE	DESCRIPTION
NEGHI	0x0A	Add large negative power
NEGLO	0x0B	Add small negative power
ZERO	0x0C	No power added
POSLO	0x0D	Add small positive power
POSHI	0x0E	Add large positive power

Table 4-6-5 lists the 25 default fuzzy rules that are hardcoded in the servo module. The 75 elements in bold type are the elements of the fuzzy rules array. These are superseded by user supplied values via a successful write of the CMD_FUZZY_CTL command or by a fuzzy rule file in eeprom. On startup the servo module will search for the fuzzy rule file #0 in the file system and load it if present. If the fuzzy rule file is not present in eeprom the default values are used.

Table 4-6-12. Default Fuzzy Rules

IF POS ERR IS	**NEGLRG**	AND SPEED IS	**NEGFST**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGLRG**	AND SPEED IS	**NEGSLW**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGLRG**	AND SPEED IS	**SAME**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGLRG**	AND SPEED IS	**POSSLW**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGLRG**	AND SPEED IS	**POSFST**	THEN OUTPUT IS	**POSLO**
IF POS ERR IS	**NEGSML**	AND SPEED IS	**NEGFST**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGSML**	AND SPEED IS	**NEGSLW**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**NEGSML**	AND SPEED IS	**SAME**	THEN OUTPUT IS	**POSLO**
IF POS ERR IS	**NEGSML**	AND SPEED IS	**POSSLW**	THEN OUTPUT IS	**ZERO**
IF POS ERR IS	**NEGSML**	AND SPEED IS	**POSFST**	THEN OUTPUT IS	**NEGLO**
IF POS ERR IS	**OK**	AND SPEED IS	**NEGFST**	THEN OUTPUT IS	**POSHI**
IF POS ERR IS	**OK**	AND SPEED IS	**NEGSLW**	THEN OUTPUT IS	**POSLO**
IF POS ERR IS	**OK**	AND SPEED IS	**SAME**	THEN OUTPUT IS	**ZERO**
IF POS ERR IS	**OK**	AND SPEED IS	**POSSLW**	THEN OUTPUT IS	**NEGLO**
IF POS ERR IS	**OK**	AND SPEED IS	**POSFST**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSSML**	AND SPEED IS	**NEGFST**	THEN OUTPUT IS	**POSLO**
IF POS ERR IS	**POSSML**	AND SPEED IS	**NEGSLW**	THEN OUTPUT IS	**ZERO**
IF POS ERR IS	**POSSML**	AND SPEED IS	**SAME**	THEN OUTPUT IS	**NEGLO**
IF POS ERR IS	**POSSML**	AND SPEED IS	**POSSLW**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSSML**	AND SPEED IS	**POSFST**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSLRG**	AND SPEED IS	**NEGFST**	THEN OUTPUT IS	**NEGLO**
IF POS ERR IS	**POSLRG**	AND SPEED IS	**NEGSLW**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSLRG**	AND SPEED IS	**SAME**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSLRG**	AND SPEED IS	**POSSLW**	THEN OUTPUT IS	**NEGHI**
IF POS ERR IS	**POSLRG**	AND SPEED IS	**POSFST**	THEN OUTPUT IS	**NEGHI**

If bit 1 of the op byte is set then the fuzzy control command reads or writes the fuzzy input and output arrays. The order of input and output data passed by the fuzzy control command is set by the defined mnemonic value (ie: the NEGLRG data is always first).

Each fuzzy rule input mnemonic is paired with a byte array that represents a 4 byte trapezoidal membership function. The geometry of the functions must conform to a 'Normal Membership Function' as described in the 'Fuzzy Logic Support' chapter of the CPU12 reference manual from Freescale. The official description of trapezoidal membership function geometry and functionality in that document is not repeated here. The FLSC validates all new trapezoidal functions for normality before committing an update.

Each fuzzy rule output mnemonic is paired with an element of the output singleton array. Validation of a new output singleton array is a simple check to confirm that the elements are increasing in value.

The FLSC module has hardcoded defaults for the input trapezoidal membership function arrays and output singleton array that are superseded by user input or a fuzzy membership function file in the eeprom. On startup the servo module will search for the fuzzy membership function file #0 in the eeprom file system and load it if present. If the fuzzy membership function file is not present in eeprom the default values are used.

Table 4-6-13. Default input membership function arrays

INPUT MNEMONIC	POINT1, POINT2, SLOPE1, SLOPE2
NEGLRG	{0x00, 0x40, 0x00, 0x04}
NEGSML	{0x00, 0x80, 0x04, 0x04}
OK	{0x40, 0xC0, 0x04, 0x04}
POSSML	{0x80, 0xFF, 0x04, 0x04}
POSLRG	{0xC0, 0xFF, 0x06, 0x00}
NEGFST	{0x00, 0x40, 0x00, 0x04}
NEGSLW	{0x00, 0x80, 0x04, 0x04}
SAME	{0x40, 0xC0, 0x04, 0x04}
POSSLW	{0x80, 0xFF, 0x04, 0x04}
POSFST	{0xC0, 0xFF, 0x06, 0x00}

Table 4-6-14. Default output singleton array

OUTPUT MNEMONIC	VALUE
NEGHI	0x00
NEGLO	0x40
ZERO	0x80
POSLO	0xC0
POSHI	0xFF

4.7 ZERO (CMD_ZERO)

The zero command sets the FLSC encoder count to zero and returns the encoder value prior to zeroing in the response message. The return value in the slave response is always SUCCESS;

The zero command has a slave write mode only. Group addressing of this command is ignored by the slave network.

Table 4-7-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x01	# data bytes (excludes header, len and lrc)
cmd	0x06	CMD_ZERO
lrc	0xF9	LRC of bytes between header and LRC

Table 4-7-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x07	# data bytes (excludes header, len and lrc)
cmd	0x06	CMD_ZERO
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
Position.hh	0x00 – 0xFF	Bits 31-24 of position before zeroing (signed 32-bit)
Position.hl	0x00 – 0xFF	Bits 23-16 of position before zeroing (signed 32-bit)
Position.lh	0x00 – 0xFF	Bits 15-8 of position before zeroing (signed 32-bit)
Position.ll	0x00 – 0xFF	Bits 7-0 of position before zeroing (signed 32-bit)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-7-3. Turnaround times

Bit Rate	Elapsed time (millisec)
125Kbps	2.875
500Kbps	.875
1Mbps	.625

4.8 STATUS (CMD_STATUS)

The status command is the primary tool for querying FLSC state and performance. This command allows the user to read servo state, errors, current position, current velocity, peak velocity, peak acceleration, limit states, limit pin values and ATD values. The write version of the command is used to clear sticky error bits.

The status command has a slave mode only. Group addressing of this command is ignored by the slave network.

4.8.1 Status Read

Table 4-8-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x07	CMD_STATUS
r/w	0x01	Write = 0; read = 1
op	0x01 – 0x1F	Bitfield to determine which optional controls the slave will return in the response (see table 4-8-9)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-8-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x05 – 0x19	# data bytes (excludes header, len and lrc)
cmd	0x07	CMD_STATUS
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
servo state	0x00 – 0x05	Servo state
errflg	0x00 – 0x3F	Sticky error bitfield
op	0x01 – 0x1F	Echo of op bitfield sent by master
current position	0x00 – 0xFF	Current position bits 31-24 (signed 32-bit) Included when bit 0 of op byte set.
current position	0x00 – 0xFF	Current position bits 23-16 (signed 32-bit) Included when bit 0 of op byte set.
current position	0x00 – 0xFF	Current position bits 15-8 (signed 32-bit) Included when bit 0 of op byte set.
current position	0x00 – 0xFF	Current position bits 7-0 (signed 32-bit) Included when bit 0 of op byte set.
current velocity	0x00 – 0xFF	Current velocity bits 31-24 (signed 32-bit) Included when bit 1 of op byte set.
current velocity	0x00 – 0xFF	Current velocity bits 23-16 (signed 32-bit) Included when bit 1 of op byte set.

current velocity	0x00 – 0xFF	Current velocity bits 15-8 (signed 32-bit) Included when bit 1 of op byte set.
current velocity	0x00 – 0xFF	Current velocity bits 7-0 (signed 32-bit) Included when bit 1 of op byte set.
peak change	0x00 – 0xFF	Peak position change bits 31-24 (unsigned 32-bit) Included when bit 2 of op byte set.
peak change	0x00 – 0xFF	Peak position change bits 23-16 (unsigned 32-bit) Included when bit 2 of op byte set.
peak change	0x00 – 0xFF	Peak position change bits 15-8 (unsigned 32-bit) Included when bit 2 of op byte set.
peak change	0x00 – 0xFF	Peak position change bits 7-0 (unsigned 32-bit) Included when bit 2 of op byte set.
peak change change	0x00 – 0xFF	Peak position changechange bits 31-24 (unsigned 32-bit) Included when bit 3 of op byte set.
peak change change	0x00 – 0xFF	Peak position changechange bits 23-16 (unsigned 32-bit) Included when bit 3 of op byte set.
peak change change	0x00 – 0xFF	Peak position changechange bits 15-8 (unsigned 32-bit) Included when bit 3 of op byte set.
peak change change	0x00 – 0xFF	Peak position changechange bits 7-0 (unsigned 32-bit) Included when bit 3 of op byte set.
limit values	0x00 – 0x0F	bitfield of limit pin values (high/low) Included when bit 4 set of op byte set.
limit states	0x00 – 0x0F	bitfield of current limit states Included when bit 4 set of op byte set.
atd value	0x00 – 0xFF	Current ATD measurement of motor current Included when bit 5 set of op byte set.
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-8-3. Turnaround times (all bytes read, all flags cleared for write)

Bit Rate	Write Elapsed time (millisec)	Read Elapsed time (millisec)
125Kbps	2.5	4.75
500Kbps	.75	1.25
1Mbps	.5	1

4.8.2 Status Write

Table 4-8-4. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x07	CMD_STATUS
r/w	0x00	Write = 0; read = 1
op	0x01 – 0x1F	Bitfield to determine which sticky error bits to clear (see table 4-8-8)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 4-8-5. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x05	# data bytes (excludes header, len and lrc)
cmd	0x07	CMD_STATUS
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
servo state	0x00 – 0x05	Servo state
errflg	0x00 – 0x3F	Sticky error bitfield
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

4.8.3 Control Options and Details

The 16-bit return value for the status command serves a different purpose than for the other commands. Normally this value pertains to the results of the action performed by the command. For the status command this return value is used to provide detail about sticky errors and system errors that are indicated by the errflg byte.

Table 4-8-6. Servo States

VALUE	STATE
0x00	NO_INIT_STATE
0x01	ACCEL_STATE
0x02	SLEW_STATE
0x03	DECEL_STATE
0x04	REVERSE_STATE
0x05	READY_STATE

The servo state byte indicates the motion state of the servo. The NO_INIT_STATE is the default state of a controller at power up if a servo control file is not present in the file system. In this state no motion control commands can be initiated. If the servo control file is present in the file system or the master has configured the servo with the servo control command, the controller transitions to the READY_STATE. Velocity and trapezoidal motions use the ACCEL_STATE, SLEW_STATE and DECEL_STATE states

to describe the 3 phases of motor motion. The REVERSE_STATE is used by trapezoidal motions when a new move control command requires the system to decelerate to zero velocity before initiating the new trajectory.

Table 4-8-7. Errflg bitfield byte

BIT	ERROR	SOURCE
0	IRQ	IRQ interrupt enabled and IRQ pin held low by external signal
1	Limit	limit event set by limit control command has been detected
2	System error	Unrecoverable system error
3	CAN error	CAN communication error
4	Position error	Illegal position detected (ie: motor polarity opposite to encoder)
5	ATD limit	motor current has exceeded the set atd limit

Table 4-8-7 lists the 'sticky' bits of the errflg bitfield byte. This byte is returned in the FLSC response for all status commands. Sticky errors are so called because they must be explicitly cleared by the master before motion can be started. These errors can only be cleared by setting the same bits in the op byte of the status write command.

The IRQ sticky bit of the errflg byte is set when the IRQ limit is enabled and a low logic signal has been detected on the IRQ pin.

If the Limit sticky bit of the errflg byte is set the source of the limit error is determined by the Limit States bitfield byte. The bit values of this byte are identical to the values of table 4-3-2. Set bits indicate limit sources that have detected a limit according to the settings of the limit control command.

If the System error sticky bit of the errflg byte is set then the retval.h and retval.l bytes contain a non-zero value that corresponds to the error codes described in section 6.

The CAN error bit of the errflg byte indicates errors detected on the CAN bus. CAN bus errors can be caused by bus contention when FLSC modules have the same ID set by their solder jumpers. CAN errors will happen if a FLSC module transmits data onto the CAN bus with a bit rate different than other modules or the master.

The Position error sticky bit is used to indicate that the encoder polarity is opposite to the motor polarity or that no encoder activity is detected after a move is started. If this error is detected then either the motor connections to the h-bridge or the A-B signals from the encoder need to be inspected.

If the ATD limit sticky bit of the errflg byte is set when the ATD limit conditions are detected.

Table 4-8-8. Op bitfield write controls

BIT	CONTROL	ACTION
0	IRQ	Clears IRQ sticky error
1	Limit	Clears Limit sticky error
2	System Error	Clears System Error sticky error
3	CAN error	Clears CAN error sticky error
4	Position error	Clears position error sticky error
5	ATD limit	Clears ATD limit sticky error

The op byte for status write uses the same bit positions as the errflg byte. Setting a bit of the op byte will clear the corresponding sticky error.

Table 4-8-9. Op bitfield read controls

BIT	CONTROL	# BYTES	DATA DESCRIPTION
0	Current position	4	Current encoder position (signed 32-bit)
1	Current velocity	4	Current velocity (signed 32-bit)
2	Peak encoder change	4	Peak position change (signed 32-bit)
3	Peak encoder change change (acc)	4	Peak position change change (unsigned 32-bit)
4	Limit	1 1	bitfield of current limit pin values bitfield of current limit states
5	ATD	1	atd measurement of motor current

The op byte for status read is used to request the return of optional read data from the FLSC slave.

Current position is the value of the encoder count. If the motor is moving quickly this value is unlikely to be exactly accurate by the time the master gets the value due to bus latencies.

Current velocity is the encoder count per second. If the motor is accelerating or decelerating quickly this value is unlikely to be exactly accurate by the time the master gets the value due to bus latencies. Though the velocity has a signed 20-bit range (see section 4.4), the returned value is formatted for signed long type calculations.

The Peak encoder change is the maximum position change in encoder counts from one servo cycle to the next. This value will never approach the boundaries of the 32-bit range however it is sized for signed long type calculations. This data is useful for determining appropriate servo rate and encoder divider values for the servo control command. Note that the peak speed of the motor is this value shifted left by the servo rate.

The Peak encoder change change is the maximum change of position change in encoder counts from one servo cycle to the next. This value will never approach the boundaries of the 32-bit range however it is sized for unsigned long type calculations. This data is useful for determining acceleration and hard stop deceleration values for the

move control command. The peak acceleration of the motor is this value twice shifted left by the servo rate.

All data related to encoder counts assumes the prior application of the encoder divider value set by the servo control command.

The 2 limit bytes are bitfields with the same bit arrangement as table 4.3.2. The first byte indicates the logic level detected at each digital limit pin. The second byte indicates the limit state for each digital limit.

The ATD byte is the voltage measured at the current sense pin of the FLSC module (see section 4.3).

4.9 STATISTICS (CMD_STATS)

The two 8-bit inputs bytes, 8-bit output byte and the servo state of the fuzzy inference routine are stored to a ring buffer at the end of every servo cycle. This ring buffer accommodates up to 1500 of the most recent iterations. If the servo rate has been set to 1KHz, the buffer will be able to record the last 1.5 seconds of motion. The purpose of these statistics is to provide data for analyzing motor behaviour during the tuning process.

When the start bit of a CMD_MOVE_CTL command is sent to begin motion, the statistics buffer is flushed and fuzzy engine I/O data collection begins. New data is added to the head of the buffer with each successive servo cycle until the servo state has entered the READY_STATE or the master has issued the CMD_STATS command. If the buffer has data, the slave responds with a maximum of two statistics. If the buffer is empty the slave responds with an INT_RESOURCE_ERR error. Master software must be designed to repeat the command until the slave returns this error.

The statistics command is slave mode only. Group addressing of this command is ignored by the slave network.

Table 4-9-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x01	# data bytes (excludes header, len and lrc)
cmd	0x08	CMD_STATS
lrc	0xF7	LRC of bytes between header and LRC

Table 4-9-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	3, 7, 11	# data bytes (excludes header, len and lrc)
cmd	0x08	CMD_STATS
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
Pos Error	0x00 – 0xFF	position error input (normalized to signed 8-bit)
Pos Change	0x00 – 0xFF	position change input (normalized to signed 8-bit)
Output	0x00 – 0xFF	Output (signed 8-bit)
Servo State	0x00 – 0x05	Servo state – see table 4-8-6
Pos Error	0x00 – 0xFF	position error input (normalized to signed 8-bit)
Pos Change	0x00 – 0xFF	position change input (normalized to signed 8-bit)
Output	0x00 – 0xFF	Output (signed 8-bit)
Servo State	0x00 – 0x05	Servo state – see table 4-8-6
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

SLEEP (CMD_SLEEP)

The sleep command is group mode only. Slave addressing of this command is ignored by the slave network.

The two sleep modes are differentiated by the wakeup mechanism. The master awakens the slave network from the CAN sleep type by placing dominant CAN bits onto the bus. The slave network is awoken from the IRQ sleep type when the IRQ line is held low.

Register and processor state are preserved by the module during sleep as long as a minimum voltage of 2.5V is available. During sleep the module is not able to recognize any external events except for the wakeup event. Changes to encoder or limit signals will have no effect on the module. If there is a chance that these signals have changed between the initiation of sleep and wakeup then re-initialization of the system should be repeated.

If the GPIO pin is configured by the CMD_SERVO_CTL command to be either GPIO_OUTPUT_LO_SLEEP or GPIO_OUTPUT_HI_SLEEP then the gpio pin of the slave will be toggled when entering sleep. This is a useful feature for switching off circuitry to reduce power consumption such as optical encoders and optical limit switches. Logic level n-channel mosfets are ideal as a low side switch for this purpose.

Table 4-10-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xAA	Start of command delimiter
len	0x01	# data bytes (excludes header, len and lrc)
cmd	0x09	CMD_SLEEP
wakeup type	0x00 – 0x01	0 = comms wakeup; 1 = IRQ wakeup
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Slave response:
None for group commands.

5. GATEWAY COMMANDS

There are 3 master commands for controlling the USB to CAN Gateway device.

Table 5-0-1. Gateway Commands

VALUE	COMMAND	FUNCTIONAL DESCRIPTION
0	CMD_ VERSION	Get gateway version information
1	CMD_ STATUS	Get status and clear sticky error
2	CMD_ BITRATE _CTL	Get/set CAN bit rate

5.1 VERSION (CMD_VERSION)

This command is used to by the master to detect the presence of a gateway device on the network. In place of the return code that normally resides in bytes 2 and 3 the slave returns the 16 bit version number of the gateway firmware.

Table 5-1-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x01	# data bytes (excludes header, len and lrc)
cmd	0x00	Version command
lrc	0xFF	LRC of bytes between header and LRC

Table 5-1-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x00	Statistics command
version high	0x00 – 0xFF	Bits 15-8 of version (major release number)
version low	0x00 – 0xFF	Bits 7-0 of version (minor release number)
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

5.2 STATUS (CMD_STATUS)

The read mode of this command polls the gateway status. Return values less than zero (error) are 'sticky' and must be cleared via a write command. The retval response of a write is the return value of the function that clears the stored error code. The last two bytes of response data is a 16-bit counter of the turnaround time of the last slave command. The gateway clock has a tick with a 125usec period. The time is the elapsed number of ticks measured between the forwarding of a master command onto the CAN bus and the reception of the slave response. A return of zero indicates that a slave response has not arrived.

Table 5-2-1. Master Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x02	# data bytes (excludes header, len and lrc)
cmd	0x01	Status command
r/w	0x00 – 0x01	Write = 0; read = 1
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 5-2-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x00	Statistics command
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
timer high	0x00 – 0xFF	Bits 15-8 of 16-bit command turnaround timer
timer low	0x00 – 0xFF	Bits 7-0 of 16-bit command turnaround timer
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

5.3 CAN BIT RATE CONTROL (CMD_BITRATE_CTL)

This command writes and reads the current CAN bit rate set inside the gateway. The regime to alter the CAN bit rate for the network is described in section 4.1.2.

Table 5-3-1. Master Write Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x02	# data bytes (excludes header, len and lrc)
cmd	0x02	CAN bit rate control command
r/w	0x00	Write = 0; read = 1
bit rate	0x00 – 0x02	Bit rates in table 4-1-4
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 5-3-2. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x00	Statistics command
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 5-3-3. Master Read Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x02	# data bytes (excludes header, len and lrc)
cmd	0x02	CAN bit rate control command
r/w	0x01	Write = 0; read = 1
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

Table 5-3-4. Slave Byte Stream

BYTE	VALUE	DESCRIPTION
header	0xCC	Start of command delimiter
len	0x03	# data bytes (excludes header, len and lrc)
cmd	0x00	Statistics command
retval high	0x00 – 0xFF	Bits 15-8 of 16-bit return value
retval low	0x00 – 0xFF	Bits 7-0 of 16-bit return value
bit rate	0x00 – 0x02	Bit rates in table 4-1-4
lrc	0x00 – 0xFF	LRC of bytes between header and LRC

6. ERROR CODES

Function return codes are signed 16-bit values. A return value of zero signifies a successful completion of a function and negative values signify an error

Error codes are grouped into blocks defined by driver or code area. The following enum defines the order of the blocks:

Table 6-1. ERROR BLOCKS

RANGE	ERROR TYPE
0xFFFF – 0xFF01	GLBOBAL ERROR
0xFF00 – 0xFE01	COMMAND ERROR
0xFE00 – 0xFD01	CAN ERROR
0xFD00 – 0xFC01	SERVO ERROR
0xFC00 – 0xFB01	EVENT ERROR
0xFB00 – 0xFA01	ATD ERROR
0xFA00 – 0Xf901	FILE ERROR
0xF900 – 0xF801	EEPROM ERROR

Each block is assigned a range of 256 possible errors such that the error code can be understood as having an upper byte for error type and a lower byte for the specific error. This is a useful distinction if the error code from an unrecoverable or CAN bus error is flashed on the module LED. The convention is that the LED will flash the value of the upper byte at a 1Hz frequency followed by a 2Hz frequency for the value of lower byte. The pattern is repeated indefinitely until the sticky bits are cleared with the status write command or a power on reset.

An example would be if two or more modules on a servo network are configured with the same id. CAN message collisions will cause the CAN_ERR_TX_BUS_OFF error to be displayed on the LEDs of the affected modules. The error code of 0xFDF9 (-0x207) will result in 2 long flashes followed by 7 short.

Table 6-2. Global Return Values

RETURN CODE	VALUE	DESCRIPTION
RUNNING	1	Asynchronous process active (no error)
SUCCESS	0	No error
FAIL	0xFFFF	Fail (general error)
INVALID_PARAM	0xFFFE	Invalid parameter error
INT_RESOURCE_ERR	0xFFFD	Internal resource error
INIT_ERR	0xFFFC	Initialization error
TIMEOUT	0xFFFB	Timeout error

Table 6-3. Command Processor Error Return Values

RETURN CODE	VALUE	DESCRIPTION
CMD_COMM_ERR	0xFF00	Communications initialization error
CMD_ZERO_LEN_ERR	0xFEFF	Zero message length
CMD_LEN_OVF_ERR	0xFEFE	Message length larger than buffer
CMD_CMD_ERR	0xFEFD	Invalid command
CMD_CHKSUM_ERR	0xFEFC	Invalid message checksum
CMD_DATA_LEN_ERR	0xFEFB	Insufficient data supplied in message
CMD_FILTER_ERR	0xFEFA	Mixed filter message (collision)
CMD_DATA_ERR	0XFEF9	Invalid data
CMD_TIMEOUT_ERR	0XFEF8	Message incomplete (truncated)

Table 6-4. CAN Error Return Values

RETURN CODE	VALUE	DESCRIPTION
CAN_ERR_INIT	0xFE00	CAN module uninitialized
CAN_ERR_HW_BUF_OVF	0xFDFF	Data overrun detected in CAN peripheral
CAN_ERR_RX_WRN	0xFDFE	RxWRN: 96 < receive error counter <= 127
CAN_ERR_RX_ERR	0xFDFD	RxERR: 127 < receive error counter <= 255
CAN_ERR_RX_BUS_OFF	0xFDFC	Bus-off: receive error counter > 255
CAN_ERR_TX_WRN	0xFDFB	TxWRN: 96 < transmit error counter <= 127
CAN_ERR_TX_ERR	0xFDFA	TxERR: 127 < transmit error counter <= 255
CAN_ERR_TX_BUS_OFF	0xFDF9	Bus-Off: transmit error counter > 255
CAN_ERR_BUF_OVF	0xFDF8	CAN data buffer overrun
CAN_ERR_BUF_EMPTY	0xFDF7	CAN data buffer empty
CAN_ERR_INVALID_ID	0xFDF6	Invalid id type (standard 11-bit only)
CAN_ERR_TIMEOUT	0xFDF5	Timeout
CAN_ERR_ABORT	0xFDF4	Failed to abort a transmission (sent)
CAN_ERR_BIT_RATE	0xFDF3	Failed to open CAN bus at set bit rate

Table 6-5. Servo Error Return Values

RETURN CODE	VALUE	DESCRIPTION
SRV_ERR_INIT	0xFD00	Servo is uninitialized
SRV_INVALID_PARAM	0xFCFF	Invalid parameter
SRV_ERR_LIMIT	0xFCFE	Limit error detected
SRV_ERR_POSITION	0xFCFD	Position error detected
SRV_ERR_VEL_ZER	0xFCFC	Velocity = 0 (trapezoidal move start)
SRV_ERR_VEL_LO	0xFCFB	Velocity too small (calculated time to goal > 32-bit)
SRV_ERR_VEL_HI	0xFCFA	Velocity > 20-bit
SRV_ERR_ACC_ZERO	0xFCF9	Acceleration = 0 (trapezoidal move start)
SRV_ERR_ACC_LO	0xFCF8	Acceleration too small (calculated ramp time > 16-bit)
SRV_ERR_ACC_HI	0xFCF7	Acceleration too large (calculated ramp time = 0)
SRV_ERR_PWM_DIR	0xFCF6	PWM direction conflicts with encoder feedback
SRV_ERR_IRQ	0xFCF5	IRQ detected
SRV_ERR_DISP_RNG	0xFCF4	Calculated distance too large (> 32-bit)
SRV_ERR_MEM_FUNC	0xFCF3	Invalid fuzzy membership function data
SRV_ERR_RULE	0xFCF2	Invalid fuzzy rule data

SRV_ERR_START_MOVE	0xFCF1	Start move when different motion mode in progress
SRV_ERR_OVF	0xFCF0	Servo engine too slow to accommodate servo rate

Table 6-6. Event (encoder) Error Return Values

RETURN CODE	VALUE	DESCRIPTION
EVT_ERR_INIT	0xFC00	Events uninitialized
EVT_INVALID_PARAM	0xFBFF	Invalid parameter
EVT_NO_EVENT	0xFBFE	No event data (encoder malfunction or disconnected)

Table 6-7. ATD Error Return Values

RETURN CODE	VALUE	DESCRIPTION
ATD_ERR_INIT	0xFB00	Channel undefined or not powered up
ATD_BUSY	0xFAFF	Conversions in process
ATD_LIMIT	0xFAFE	ATD value greater than set limit

Table 6-8. File System Error Return Values

RETURN CODE	VALUE	DESCRIPTION
FIL_ERR_INIT	0xFA00	File operation invoked on uninitialized file system
FIL_ERR_INVALID_PARAM	0xF9FF	Invalid file number or operation
FIL_ERR_NOT_FOUND	0xF9FE	File search fails to find file by name
FIL_ERR_NAME	0xF9FD	Invalid file name
FIL_ERR_SIZE	0xF9FC	File size doesn't conform to set size
FIL_ERR_FULL	0xF9FB	Insufficient room in fs to write a file
FIL_ERR_CHKSUM	0xF9FA	Checksum of file data invalid
FIL_ERR_COMP	0xF9F9	File system compression failure
FIL_ERR_CORRUPT	0xF9F8	File system data is corrupted
FIL_ERR_BUF_OVF	0xF9F7	File operation buffer full

Table 6-9. EEPROM driver error return values

RETURN CODE	VALUE	DESCRIPTION
EEP_ERR_INIT	0xF900	Eeprom driver not initialized
EEP_ERR_INVALID_PARAM	0xF8FF	Invalid eeprom parameter
EEP_ERR_ADDR	0xF8FE	Invalid eeprom address
EEP_ERR_ACCESS	0xF8FD	Illegal access to eeprom memory
EEP_ERR_PROT_VIOL	0xF8FC	Operation on a protected area
EEP_ERR_MERASE	0xF8FB	Failure to mass erase
EEP_ERR_TIMEOUT	0xF8FA	Timeout on eeprom operation