1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

DMC60C CAN Protocol Guide

Revised October 24, 2018
This manual applies to the DMC60C rev. E.1 with application firmware version 1.23 or newer and bootloader 1.9 or newer

Overview

The DMC60C follows the CAN 2.0B Active Specification, which supports both standard and extended data frames. Standard data frames include an 11-bit message identifier, which will be referred to as a Standard Identifier (SID) in this documentation. Extended data frames include the 11-bit Standard Identifier and an additional 18-bit Extended Identifier (EID), giving these types of frames a 29-bit long message identifier. The DMC60C makes exclusive use of 29-bit message identifiers and does not accept nor transmit any messages containing 11-bit message identifiers.

Byte 3								Byte 2								Byte 1								Byte 0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			Device Type					Manufacturer								API										Device Number					
			Standard Identifier											Extended Identifier																	
			10	9	8	7	6	5	4	3	2	1	0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 1: Message Identifier Fields

Value	Encoding
0	Broadcast Messages
1	Robot Controller
2	Motor Controller
3	Relay Controller
4	Gyro Sensor
5	Accelerometer Sensor
6	Ultrasonic Sensor
7	Gear Tooth Sensor
8-31	Reserved

Table 2: Device Type Encodings

Value	Encoding
0	Broadcast Messages
1	National Instruments
2	Texas Instruments (Stellaris)
3	DEKA
4	Cross The Road Electronics
5	UNKNOWN
6	Digilent
7-255	Reserved

Table 3: Manufacturer Encodings

Enumeration and Device Discovery

Enumeration is the process of discovering the devices attached to the CAN bus. This process is initiated by transmitting msgidEnum (0x00000240) with the Device Number field of the message identifier set to 0. All devices that are active on the CAN bus will transmit one or more messages in response to an enumeration request. Each Digilent device that is present on the bus responds by transmitting two response packets (ENUMRSP0 and ENUMRSP1) with msgidEnumResp (0x0206F000). If the device has had a Device Number assigned to it then it's Device Number will be appended to the message identifier. Otherwise, the Device Number field of the message identifier will be set to 0.

Since there is a possibility of more than one device responding with the same Device Number it is necessary to include an additional piece of information to distinguish these devices from one another. This is done by having each device assign itself a Session ID (sessid) each time it connects to the CAN bus. The Session ID is included in both ENUMRSP0 and ENUMRSP1 packets, and allows the Robot Controller (or host) to identify when there is more than one device present on the bus that has the same Device Number assigned to it. The Session ID can then be used in conjunction with a device's present Device Number to construct Vendor Commands. Vendor commands allow the Robot Controller to retrieve additional information from a device, set a device's Device Number, perform firmware upgrades, and perform general device configuration.

When a Digilent device receives an enumeration request it will delay its response to that request by 0.5 to 63.5 milliseconds. This is done to prevent the bus from being flooded with messages and reduce frame errors. How long each device delays its response will depend on whether it's using the default Device Number (0). If a device is using the default Device Number, then the amount of delay is pseudo randomly generated using the device's current Session ID. If the device has been assigned a non-default Device Number (1-63) then the amount of delay is equal to Device Number milliseconds.

A typical enumeration process may consist of the following steps:

1. Transmit msgidEnum (0x00000240) with the Device Number field of the message identifier set to 0.
2. Receive ENUMRSP0 and ENUMRSP1 packets for up to 100 milliseconds.
3. Organize ENUMRSP0 and ENUMRSP1 packets by Device Number and Session ID to determine how many devices are present on the bus.
4. Request Device Descriptors from any device of interest.

The ENUMRSP0 and ENUMRSP1 packets returned by a device provide several useful pieces of information about the device, including its Product Identifier, Application Firmware revision, Bootloader Firmware Revision, and the type of firmware that the device is currently executing (Application, Bootloader, or Auxiliary Bootloader). Additional information can be retrieved from a device by issuing the *vcmdGetDescriptors* Vendor Command, which will cause the device to transmit its Device Descriptors. The ENUMRSP0 and ENUMRSP1 packets are described below. Additional information regarding Device Descriptors and Vendor Commands can be found in their respective sections.

ENUMRSP0 Data Structure

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
pdid				sessid	

sessid CAN bus Session ID.
pdid Unsigned 32-bit Product Identifier.

ENUMRSP1 Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
fwverBoot		fwverApp		flgsEnum		sessid	

sessid CAN bus Session ID.

flgsEnum

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rsv														imgtyp	

imgtyp Type of firmware currently executing.
0 = Application
1 = Bootloader
2 = Auxiliary Bootloader
rsv Reserved for future use.

fwverApp Application firmware revision. Device will report 0xFFFF if application isn't present.
fwverBoot Bootloader firmware revision.

Device Descriptors

The DMC60C stores several string descriptor fields in a dedicated section of flash should not be overwritten as part of an infield Application or Bootloader firmware update. These fields provide useful information about a device, such as its name, date of manufacture, and hardware revision. The Robot Controller (or host) may obtain a device's string descriptors by using the *vcmdGetDescriptors* Vendor Command. The device returns its string descriptors using a field separated binary stream. The first byte of each field serves as a field identifier (idfld). The second byte of each field is a count of the number of bytes (cb), or characters, that make up the string corresponding to the current field, which immediately follow the byte count. The following is an example of an abbreviated binary stream returned in response to the *vcmdGetDescriptors* Vendor Command:

BYTE #	0	1	2	3	4	5	6	7
DATA	idfldSzDevName	6	'D'	'M'	'C'	'6'	'0'	'C'

BYTE #	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DATA	idfldSzProdName	15	'D'	'i'	'g'	'i'	'l'	'e'	'n'	't'	' '	'D'	'M'	'C'	'6'	'0'	'C'

BYTE #	25	26	27	28	29	30	31	32	33	34	35	36	37	38
DATA	idfldSzSN	13	'2'	'1'	'0'	'3'	'3'	'4'	'3'	'A'	'7'	'9'	'F'	'2'

Each string field can be individually assigned using commands described in the Vendor Commands section.

The table below defines the field identifiers used to identify each type of descriptor and provides an overview of the string fields that are available. A detailed description of each string field follows below.

Field Identifier	Value	Description	Maximum Number of Characters	Default Value
idfldNone	0	NA	NA	NA
idfldSzDevName	1	Device Name String	28	"Motor Controller"
idfldSzManName	2	Manufacturer Name String	28	"Digilent"
idfldSzProdName	3	Product Name String	28	"Digilent DMC60C"
idfldSzManDate	4	Manufacture Date String	20	"UNKNOWN"
idfldSzHardWareVer	5	Hardware Version Number String	8	"UNKNOWN"
idfldSzSN	6	Serial Number String	12	"UNKNOWN"

Note: the maximum number of characters does NOT include the zero terminator.
Note: Default Value is the value that will be returned for a field if no value has been set or the EEPROM has been corrupted. All fields should have a value assigned during the manufacturing test.

Device Name String
The Device Name String is intended to be a user assignable device name. It can be set using the *vcmdSetDevName* Vendor Command. During the manufacturing test a default value of "DMC60C" will be assigned to this field.

Manufacturer Name String
The Manufacturer Name String lists the manufacturer of the product. This is not meant to be a user assignable field. However, it can be set using the *vcmdSetManName* Vendor Command. During the manufacturing test a default value of "Digilent" will be assigned to this field.

Product Name String
The Product Name String provide a descriptive name that corresponds to the device. This is not meant to be a user assignable field. However, it can be set using the *vcmdSetProdName* Vendor Command. During the manufacturing test a default value of "Digilent DMC60C" will be assigned to this field.

Manufacture Date String
The Manufacture Date string lists the date that the product was manufactured. The date string should be of the format "mm/dd/yy" where "mm" corresponds to the month (1 – 12), "dd" corresponds to the day of the month (1 – 31), and "yy" corresponds to the year. For example, if the product was manufactured on January 16 of 2018 then a value of "01/16/18" should be assigned. This is not meant to be a user assignable field. However, it can be set using the *vcmdSetManDate* Vendor Command. During the manufacturing test the current date will be assigned to this field.

Hardware Version Number String
The Hardware Version Number string lists the revision of the current PCB assembly contained within the product. The hardware version number string should be of the format "x.y" where "x" is the major revision letter and "y" is the minor revision number. For example, if the current revision of the PCB assembly is "E.0" then a value of "E.0" should be assigned. This is not meant to be a user assignable field. However, it can be set using the *vcmdSetHardwareVer* Vendor Command. During the manufacturing test the current PCB assembly revision will be assigned to this field.

Serial Number String
The Serial Number String is a 12-digit hexadecimal number that is intended to be unique for every device manufactured. The first 6 digits of the serial number string are fixed to "210334" while the remaining 6 digits are assigned by the contract manufacturer and should be unique to each device that's manufactured. This is not meant to be a user assignable field. However, it can be set using the *vcmdSetSN* Vendor Command. During the manufacturing test the device's serial number will be assigned to this field.

Vendor Commands

Overview

Vendor commands are used for performing general device configuration and maintenance. They are used to set string descriptors, retrieve string descriptors, assign device numbers, and perform firmware upgrades. All vendor commands consist of a command packet, a status packet, and optionally include bulk data transfer to or from the device.

Commands that do not involve transmitting or receiving bulk data are initiated by transmitting a *VENDORCMD* packet using identifier *msgidVendorCmd*. Upon processing this command the device will respond by transmitting a *VENDORSTS* packet using message identifier *msgidVendorSts*. The *cerc* field of the *VENDORSTS* packet will contain an error code corresponding to the command and the byte count field, *cb*, will be set to 0.

Commands that involve transferring bulk data to the device will include a byte count in the appropriate parameter of the *VENDORCMD* packet and will transmit the packet using message identifier *msgidVendorCmd*. Bulk data will then be transmitted to the device using message identifier *msgidVendorDout*. Once all data has been transmitted the device will respond with a *VENDORSTS* packet containing an error code and the number of bytes that were successfully received.

Commands that involve receiving bulk data from a device are initiated by transmitting a *VENDORCMD* packet using identifier *msgidVendorCmd*. Upon processing this command the device will respond with a *VENDORSTS* packet containing an error code and the number of bytes that the device intends to transmit in response to the command. If the command is accepted, then the device will begin transmitting bulk data using message identifier *msgidVendorDin*. The device will continue transmitting bulk data until all data has been transmitted, and error has occurred, or a new command has been received.

Message Identifiers, Error Codes, and Data Structures

Vendor Command Message Identifiers

Message Identifier	Value
msgidVendorCmd	0x0206FC00
msgidVendorDout	0x0206FC40
msgidVendorDin	0x0206FC80
msgidVendorSts	0x0206FCC0

Note: All message identifiers transmitted as part of a vendor command should include a Device Number in the lower 6 bits of the extended identifier.

Vendor Command Error Codes

Error	Value	Description
cercNoError	0	No error has occurred
cercNotSupported	1	The specified command is not supported by the current firmware image
cercBadParameter	2	One or more parameter specified is invalid for the specified command
cercDataRcvMore	3	The device received more data than was specified in the command
cercInBootloader	4	The device was instructed to the enter the bootloader and the bootloader is running
cercCrcMismatch	5	The CRC computed for the flash page does not match the CRC received from the host
cercFlashWriteFailed	6	The device either failed to erase or write the specified flash page
cercAckReset	7	The device was instructed to reset and now that reset has completed
cercTestPassed	8	The test that was run in response to the received vendor command passed
cercTestFailed	9	The test that was run in response to the received vendor command failed

Test Error Codes

Error	Value	Description
tercNoError	0	No error has occurred
tercUserAN1	1	Pin AIN1 failed the test specified by the vendor command
tercFwdLimit	2	Pin FWDLIM failed the test specified by the vendor command
tercRevLimit	3	Pin REVLIM failed the test specified by the vendor command
tercQEA	4	Pin QEA failed the test specified by the vendor command
tercQEB	5	Pin QEB failed the test specified by the vendor command
tercQEIdx	6	Pin QEIDX failed the test specified by the vendor command

VENDORCMD Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
PARAM2		PARAM1		cmd		sessid	

sessid CAN bus Session ID.
cmd Vendor command.
PARAM1 First command parameter. Meaning varies depending on the command.
PARAM2 Second command parameter. Meaning varies depending on the command.

VENDORSTS Data Structure

Byte 3	Byte 2	Byte 1	Byte 0
cb		cerc	

cerc Command error code returned in response to the most recent command.
cb Count of bytes to be returned through DTI endpoint.

FWVERRSP Data Structure

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
fwverBoot		fwverApp		flgsFwr	

flgsFwr

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rsv														imgtyp	

imgtyp Type of firmware currently executing.
0 = Application
1 = Bootloader
2 = Auxiliary Bootloader

rsv Reserved for future use.

fwverApp Application firmware revision. Device will report 0xFFFF if application isn't present.
fwverBoot Bootloader firmware revision.

Commands

vcmdSetDevNumber **0x01**

Parameters

PARAM1 Device number to assign. Valid values are 1 – 63.
PARAM2 0

Set the device number to the specified value. The device number is specified in the *PARAM1* field of the *VENDORCMD* packet, which is transferred with *msgidVendorCmd*. After the device has received and processed the command it will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If assignment of the new device number was successful, then the new device number will be present in the lower 6-bits of the EID field of the *VENDORSTS* packet and the *cerc* field will be set to *cercNoError*. If an error occurred, then the lower 6-bits of the EID field of the *VENDORSTS* packet will contain the device's original device number and the *cerc* field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetDevName **0x02**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.
PARAM2 0

Set the device's device name string to the specified value. The device name string may be up to 64 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the device name string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetManName **0x03**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.
PARAM2 0

Set the device's manufacturer name string to the specified value. The manufacturer name string may be up to 28 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the manufacturer name string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetProdName **0x04**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.

PARAM2 0

Set the device's product name string to the specified value. The product name string may be up to 28 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the product name string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetManDate **0x05**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.

PARAM2 0

Set the device's manufacture date string to the specified value. The manufacture date string may be up to 20 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the manufacture date string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetHardwareVer **0x06**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.

PARAM2 0

Set the device's hardware version string to the specified value. The hardware version string may be up to 8 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the hardware version string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdSetSN **0x07**

Parameters

PARAM1 Number of characters in the string, including the zero terminator.

PARAM2 0

Set the device's serial number string to the specified value. The serial number string may be up to 8 characters long, not including the zero terminator, and must be zero terminated. The specified string should be transferred to the device by sending one or more messages with *msgidVendorDout* immediately following the vendor command. Once all characters have been transferred the device will process the command and send a *VENDORSTS* packet with *msgidVendorSts*. If the serial number string was successfully written to flash memory, then the *cerc* field of the *VENDORSTS* packet will contain cercNoError. If an error occurred, then the cerc field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdFlashLEDS **0x50**

Parameters

PARAM1 0

PARAM2 0

Instruct the device to cycle its LEDs in a rainbow-like pattern. The corner LEDs will continuously toggle between red, orange, yellow, green, blue, fuchsia, and cyan for 5 seconds and then revert to their previous state. The device will respond with a *VENDORSTS* packet containing *msgidVendorSts* immediately after processing the command. The *cerc* field of the *VENDORSTS* packet will contain *cercNoError* and the *cb* field will be set to 0.

vcmdGetDescriptors **0x60**

Parameters

PARAM1 0

PARAM2 0

Retrieve the device's string descriptors. After the device has received and processed the command it will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If no error occurred, then the *cerc* field of the *VENDORSTS* packet will contain *cercNoError* and the *cb* field will contain the total number of bytes to be received from the device. The device will then begin transmitting messages with *msgidVendorDin*, each containing up to 8 bytes of data. The device will continue to transmit messages with *msgidVendorDin* until *cb* bytes have been transferred, a new command has been received, or an error occurs. The device will NOT transmit a *VENDORSTS* packet at the end of the transfer. The string descriptors are returned as a field separated binary stream. The first byte of each field serves as a field identifier (*idfld*). The second byte of each field is a count of the number of bytes (*cb*), or characters, that make up the string corresponding to the current field, which immediately follow the byte count. Please note that the string descriptors returned in response to this command do NOT include a zero-terminator. The robot controller, or host, must parse the binary stream into the applicable descriptor strings and add zero-terminators as needed.

vcmdGetFwver **0x61**

Parameters

PARAM1 0

PARAM2 0

Retrieve the device's firmware version information. This includes the application firmware revision, bootloader firmware revision, and the type of firmware image that is currently running (bootloader, application, or auxiliary bootloader). After the device has received and processed the command it will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If no error occurred, then the *cerc* field of the *VENDORSTS* packet will contain *cercNoError* and the *cb* field will contain the total number of bytes to be received from the device. The device will then transmit a FWVERRSP packet with *msgidVendorDin*, containing *cb* bytes of data. The device will NOT transmit a *VENDORSTS* packet at the end of the transfer. Please note that the fwverApp field of the FWVERRSP packet will contain 0xFFFF if no application firmware is present in the device's flash memory.

vcmdGetFlashSeqnum **0x62**

Parameters

PARAM1 0

PARAM2 0

Retrieve the device's flash sequence number. The flash sequence number is a 32-bit unsigned value that represents sequence number corresponding to the most recently written EEPROM section. Each time the EEPROM section is written to flash the sequence number is incremented. The sequence number, along with knowledge of the memory map, and the flash endurance, can be used to determine the wear level of the flash memory that's used for storing nonvolatile configuration parameters. After the device has received and processed the *vcmdGetFlashSeqnum* command it will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If no error occurred, then the *cerc* field of the *VENDORSTS* packet will contain *cercNoError* and the *cb* field will contain the total number of bytes to be received from the device. The device will then transmit a FWVERRSP packet with *msgidVendorDin*, containing *cb* bytes of data. The device will NOT transmit a *VENDORSTS* packet at the end of the transfer.

vcmdEnterBootloader **0xF0**

Parameters

PARAM1 0

PARAM2 0

Instruct the device to terminate any application firmware that may be running and jump back into the bootloader. After the device has received and processed the command it will save the current session ID, set a flag in persistent memory telling it to remain in the bootloader, and perform a software reset. Once the device is executing the bootloader firmware it will then respond with a *VENDORSTS* packet containing *msgidVendorSts*. The *cerc* field of the *VENDORSTS* packet will contain *cercInBootloader* and the *cb* field will be set to 0. If the device receives this command while it's already executing the bootloader firmware, then no reset will occur, and the device will respond with the same status message.

vcmdSoftReset **0xF1**

Parameters

PARAM1 0

PARAM2 0

Instruct the device to perform a software reset. After the device has received and processed the command it will save the current session ID, set a flag in persistent memory telling it to send a status packet once the reset has completed, and perform a software reset. After the device performs the reset and begins executing the application firmware (if present) then it will respond with *VENDORSTS* packet containing *msgidVendorSts*. The *cerc* field of the *VENDORSTS* packet will contain *cercAckReset* and the *cb* field will be set to 0. If the device receives this command and no application firmware is present, then it will still perform the software reset, but no *VENDORSTS* packet will be sent.

vcmdEraseWriteFlashPage **0xF2**

Parameters

PARAM1 Image Type (Application = 0, Bootloader = 1, Auxiliary Bootloader = 2)

PARAM2 Flash Page Number

Erase and then program the specified page of the device's flash memory. Each page of flash memory contains 1024 instructions, with each instruction being 24-bits wide. However, the device expects each instruction to be provided as a 32-bit DWORD with the most significant 8-bits serving the function of padding. Additionally, the device expects a 4-byte CRC32 to immediately follow the 1024 instructions. After the device receives the command it will verify that the specified image type can be programmed by the currently executing firmware image and that the specified page number is in the valid range for the specified image type. If the specified image type cannot be programmed by the current firmware image, then the device will transmit a *VENDORSTS* packet with *msgidVendorSts* and the *cerc* field of the *VENDORSTS* packet will contain *cercBadParameter* (Bootloader or Auxiliary Bootloader firmware executing) or *cercNotSupported* (Application firmware executing). If both the image type and flash page number are valid then the device will configure the DTO endpoint to receive the flash page and wait for the data transfer to complete. The flash page should then be transferred to the device by sending messages with *msgidVendorDout* immediately following the vendor command. Once the entire flash page has been received the device will compute a CRC32 of the page data, verify that it matches the CRC32 that it received, erase the page, and then write the new page to flash memory. If the entire flash page was successfully written, then then the *cerc* field of the *VENDORSTS* packet will contain *cercNoError*. If an error occurred, then the *cerc* field will contain an applicable error code. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the command was successful.

vcmdJ1ShortTest **0xF3**

Parameters

PARAM1 0

PARAM2 0

Instruct the device to run the J1 Short Circuit Test. Once the test completes the device will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If no short circuits were detected, then the *cerc* field of the *VENDORSTS* packet will contain *cercTestPassed*. If one or more short circuits were detected, then the least significant byte of the *cerc* field of the *VENDORSTS* packet will contain *cercTestFailed* and the most significant byte of the *cerc* field will contain a test error code corresponding to the last pin to fail the test. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the test passed.

vcmdJ1OpenTest **0xF4**

Parameters

PARAM1 0

PARAM2 0

Instruct the device to run the J1 Open Circuit Test. Once the test completes the device will respond with a *VENDORSTS* packet containing *msgidVendorSts*. If no open circuits were detected, then the *cerc* field of the *VENDORSTS* packet will contain *cercTestPassed*. If one or more open circuits were detected, then the least significant byte of the *cerc* field of the *VENDORSTS* packet will contain *cercTestFailed* and the most significant byte of the *cerc* field will contain a test error code corresponding to the last pin to fail the test. The *cb* field of the *VENDORSTS* packet will be 0 regardless of whether the test passed. For this test to pass a loopback fixture that connects pins 3 to 4, 5 to 7, and 8 to 9 must be connected to header J1 of the device.

Firmware Updates

The DMC60C contains 129KB of internal flash. The internal flash is divided into 43 pages, each of which contain 1024 word addressable blocks. Each address of program memory corresponds to a 16-bit lower word and a 16-bit upper word. The upper byte of the upper word is unimplemented and always reads 0, meaning that each program instruction is effectively 24-bits. All program memory addresses are word aligned to the lower word and the program counter is always executed or decremented by two during execution.

Program Memory Organization

Bit Number	31 24	23 16	15 8	7 0
PC Address	Most Significant Word		Least Significant Word	
0x000000	Phantom Byte	Program Memory		
0x000002	Phantom Byte	Program Memory		
0x000004	Phantom Byte	Program Memory		
0x000006	Phantom Byte	Program Memory		

Note: phantom byte always reads as '0'

The DMC60C's internal flash has been divided into multiple sections, allowing it to contain both bootloader and application firmware images, as well as provide non-volatile storage of various configuration parameters and string descriptors. The table below details the memory map utilized by the DMC60C.

DMC60C Memory Map

Flash Section	Byte Address	Page Number	Byte Length
GOTO Instruction	0x000000	0	0x2
Reset Address	0x000002	0	0x2
Interrupt Vector Table	0x000004 0x0001FE	0	0x1FC
Application Firmware	0x000200 0x00F7FC	0 30	0xF5FE
Application Firmware Version	0x00F7FE	30	0x2
Configuration Parameters And String Descriptors	0x00F800 0x0117FE	31 34	0x2000
Bootloader Firmware	0x011800 0x0157E8	35 42	0x3FEA
Bootloader Firmware Version	0x0157EA	42	0x2
Flash Configuration Bytes	0x0157EC 0x0157FE	42	0x14

The DMC60C comes pre-loaded with both bootloader and application firmware images. At power on or software reset the device will execute the GOTO Instruction and jump to the location specified in the Reset Address section of the flash memory. This will result in the program counter moving to the beginning of the bootloader firmware section. The processor will then begin executing the bootloader firmware.

During the initialization process the bootloader reads the *fStayInBootloader* flag, which is stored in persistent memory. If the flag is set then the bootloader will continue executing until the next power on or software reset occurs, and it will make no attempt to load the application firmware. If the flag is cleared, then the bootloader will read the *Application Firmware Version* and check to see if a valid application firmware image is present. When the *Application Firmware* Version is not 0xFFFF, which is the value set during a page erase operation, the bootloader assumes the application image is valid, frees up any resources that were in use, and then jumps to the beginning of the application firmware section. When the *Application Firmware Version* is 0xFFFF the device will continue executing the bootloader until the next power on or software reset occurs.

The bootloader can perform Run-time Self Programming (RTSP) of the device's flash memory. When the bootloader is executing, a new application firmware can be transferred over the CAN bus and programmed into flash memory one page at a time using the *vcmdEraseWriteFlashPage* vendor command.

A Robot Controller (or host) that wishes to update the application Firmware of a DMC60C device must first read in the hex file containing the firmware and parse it into pages of 4096 bytes (1024 instructions, with the MSB of the MSW padded with 0xFF). A CRC32 must be computed for each page that's present in the hex file and appended to the end of the page. When the device receives the new page, it will compute a CRC32 of the first 4096 bytes and then compare it to that of the CRC32 that received. If the CRC matches, then the device will proceed to erase the flash section. Otherwise, it will return a CRC error and abort the operation.

A hex file containing an application firmware image may contain pages that are outside of the flash section that has been reserved for application images. If the hex file contains data for pages beyond page 30 then the Robot Controller (or host) should ignore these pages and not transmit them to the device as part of a firmware update. When the bootloader receives a *vcmdEraseWriteFlashPage* command it checks the page number specified in the PARAM2 field and if it's not in a range that's valid for an application firmware or auxiliary bootloader firmware (0 to 30), then it will return a status packet containing *cercBadParameter* and abort the operation. A key ramification of this is that the bootloader and application must use the same config bits, and if a developer wants to change the config bits utilized by the application then it's also necessary to update the bootloader.

Due to the way the bootloader determines the validity of an application firmware image that's presently stored in flash it is necessary to set the application firmware version number to 0xFFFF at the beginning of a firmware update and then write the new version number to flash after all other data has been successfully programmed. After the DMC60C has received a page of data through the *vcmdEraseWriteFlashPage* vendor command and verified that the CRC32 matches it looks at the page number specified in the *PARAM2* field of the command. When page number 30 is specified the bootloader firmware saves the value of firmware version number in a temporary variable, erases the page, and then writes the first 4092 bytes of the page to flash. When page number 0 is received the bootloader firmware replaces the reset address specified in the page data with the address of the beginning of the bootloader firmware image, erases page 0, programs page 0, and then writes the firmware version number to the last DWORD of page 30. Therefore a Robot Controller (or host) that initiates an application firmware update should always write page 30 first and page 0 last. Any other pages that are in the valid range for an application firmware may be written in any order.

Recommended Application Firmware Update Steps:

1. Read and parse the hex file containing the firmware image into pages.

2. Send *vcmdEnterBootloader* to enter the bootloader and wait for a status packet containing *cercInBootloader*, which will confirm that the bootloader is now running.
3. Erase and program flash pages one at a time using the *vcmdEraseWriteFlashPage* command. Start on page 30 (the last page of the application firmware section) and work backwards to page 0, programming each page that's present in hex file.
4. Send *vcmdSoftReset* which should cause the bootloader to reset and then load the application firmware. A status packet containing *cercAckReset* should be sent by the device shortly after the command is processed.
5. Send *vcmdGetFwver* to retrieve the application firmware version number. Verify that it matches that of the newly programmed image.

Configuration Parameters

Configuration parameters are used for configuring various device settings, including the limit switches, and closed loop control constant. Most of the parameters that may be set are stored in a reserved section of the DMC60's flash memory and are preserved across power cycles. At power on the DMC60's application firmware reads the non-volatile parameters from the flash into RAM variables that are maintained while the device is operating.

The Robot Controller (or host) may set a parameter by transmitting a *PARAMSET* packet with identifier *msgidParamSet*. The *PARAMSET* packet should contain the Session ID of the target device in the *sessid* field, the Parameter Identifier in the *paramid* field, and the desired value in the value field. If the parameter is supported by the device, then the applicable variable will be updated in RAM immediately and may also be immediately written to flash if no parameter has been updated in the past 15 seconds. If a parameter has been updated or written to flash in the past 15 seconds then the flash write will be deferred until no parameters have been updated for 15 seconds, and then the write will take place. This is necessary to reduce flash wear. Once a parameter has been updated in RAM the device will respond with a *PARAMRESP* packet with identifier *msgidParamResp*. The *paramid* field of the *PARAMREQ* packet should contain the Parameter Identifier of the most recently set or requested parameter and the *value* field should contain the value that is currently set for that parameter. If no error occurred, then the *perc* field will contain *percNoError*. If the device does not support the specified parameter or an error occurred, then the *perc* field will contain an applicable error code and the *value* field will be set to 0. If the *sessid* field does not match the device's current Session ID then the device will ignore the *PARAMSET* or *PARAMREQ* packet and will not respond with a *PARAMRESP* packet.

The Robot Controller (or host) may request the value of a parameter by transmitting a PARAMREQ packet with identifier msgidParamReq. The PARAMREQ packet should contain the Parameter Identifier in the paramid field. The device will respond with the value of a parameter a *PARAMRESP* packet with identifier *msgidParamResp*. The *paramid* field of the *PARAMREQ* packet should contain the Parameter Identifier of the most recently requested parameter and the *value* field should contain the value that is currently set for that parameter. If no error occurred, then the *perc* field will contain *percNoError*. If the device does not support the specified parameter or an error occurred, then the *perc* field will contain an applicable error code and the *value* field will be set to 0.

Configuration Command Message Identifiers

Message Identifier	Value
msgidParamReq	0x02061800
msgidParamResp	0x02061840
msgidParamSet	0x02061880

Note: All message identifiers transmitted as part of a configuration command should include a Device Number in the lower 6 bits of the extended identifier.

Configuration Command Error Codes

Error	Value	Description
percNoError	0	No error has occurred
percBadParameter	1	The specified parameter is not supported
percBadValue	2	The specified value is invalid for the parameter being set

PARAMSET Data Structure

Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
value				paramid	sessid	

sessid CAN bus Session ID.
paramid Configuration parameter to set.
value Value to set for the specified configuration parameter. Meaning varies with parameter.

PARAMREQ Data Structure

Byte 2	Byte 1	Byte 0
paramid	sessid	

sessid CAN bus Session ID.
paramid Configuration parameter to request.

PARAMRESP Data Structure

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
perc	value				paramid

paramid Configuration parameter that was most recently set or requested.
value Value set for the specified parameter.
perc Error code returned in response to the most recent Parameter Set or Parameter Request command.

Configuration Parameter Summary

Parameter	ID	Description	Default Configuration	Saved in Flash
paramLimitSwtFwdNormClosed	1	Forward limit switch type	Normally Open	YES
paramLimitSwtFwdDisabled	2	Forward limit switch enable state	Enabled	YES
paramLimitSwtRevNormClosed	3	Reverse limit switch type	Normally Open	YES
paramLimitSwtRevDisabled	4	Reverse limit switch enable state	Enabled	YES
paramLimitSoftFwdThreshold	5	Forward soft limit threshold	0	YES
paramLimitSoftFwdEnable	6	Forward soft limit enable state	Disabled	YES
paramLimitSoftRevThreshold	7	Reverse soft limit threshold	0	YES
paramLimitSoftRevEnable	8	Reverse soft limit enable state	Disabled	YES
paramAdcCurrentMultiplier	9	Multiplier used to convert ADC readings into current measurements	0x00000816	YES
paramClosedLoopPGainSlot0	10	Slot 0 closed loop control proportional gain	0	YES
paramClosedLoopIGainSlot0	11	Slot 0 closed loop control integral gain	0	YES
paramClosedLoopDGainSlot0	12	Slot 0 closed loop control derivative gain	0	YES
paramClosedLoopIZoneSlot0	13	Slot 0 closed loop control integral accumulator limit	0 (disabled)	YES

paramClosedLoopFGainSlot0	14	Slot 0 closed loop control feed-forward gain	0	YES
paramClosedLoopAllowableErrorSlot0	15	Slot 0 closed loop control allowable closed loop error	0	YES
paramClosedLoopRampRateSlot0	16	Slot 0 closed loop control ramp rate	0 (disabled)	YES
paramClosedLoopFwdMaxSlot0	17	Slot 0 closed loop control maximum forward duty cycle	32767	YES
paramClosedLoopRevMaxSlot0	18	Slot 0 closed loop control maximum reverse duty cycle	-32768	YES
paramClosedLoopFwdNominalSlot0	19	Slot 0 closed loop control forward nominal duty cycle	0	YES
paramClosedLoopRevNominalSlot0	20	Slot 0 closed loop control reverse nominal duty cycle	0	YES
paramClosedLoopPGainSlot1	21	Slot 1 closed loop control proportional gain	0	YES
paramClosedLoopIGainSlot1	22	Slot 1 closed loop control integral gain	0	YES
paramClosedLoopDGainSlot1	23	Slot 1 closed loop control derivative gain	0	YES
paramClosedLoopIZoneSlot1	24	Slot 1 closed loop control integral accumulator limit	0 (disabled)	YES
paramClosedLoopFGainSlot1	25	Slot 1 closed loop control feed-forward gain	0	YES
paramClosedLoopAllowableErrorSlot1	26	Slot 1 closed loop control allowable closed loop error	0	YES
paramClosedLoopRampRateSlot1	27	Slot 1 closed loop control ramp rate	0 (disabled)	YES
paramClosedLoopFwdMaxSlot1	28	Slot 1 closed loop control maximum forward duty cycle	32767	YES
paramClosedLoopRevMaxSlot1	29	Slot 1 closed loop control maximum reverse duty cycle	-32768	YES
paramClosedLoopFwdNominalSlot1	30	Slot 1 closed loop control forward nominal duty cycle	0	YES
paramClosedLoopRevNominalSlot1	31	Slot 1 closed loop control reverse nominal duty cycle	0	YES
paramCurrentLimitPGain	32	Proportional gain constant for current limiting	0x00640000	YES
paramCurrentLimitIGain	33	Integral gain constant for current limiting	0x003C0000	YES
paramCurrentLimitDGain	34	Derivative gain constant for current limiting	0	YES
paramCurrentLimitIZone	35	Integral accumulator limit for current limiting	0x014CC888	YES
paramCurrentLimitFGain	36	Feed-forward gain constant for current limiting	0	YES
paramEncoderPosition	41	Current Encoder Position	0 at power on	NO
paramClearPositionOnIndex	42	Clear Encoder Position based on the Index Pin	0 (disabled)	NO
paramClearPositionOnFwdLimit	43	Clear Encoder Position based on the Forward Limit Switch input	0 (disabled)	NO
paramClearPositionOnRevLimit	44	Clear Encoder Position based on the Reverse Limit Switch Input	0 (disabled)	NO
paramIndexActiveEdge	45	Index pin active edge	0 (falling)	NO
paramActiveFaults	51	Active Faults	0 at power on	NO
paramStickyFaults	52	Sticky Faults	0	YES
paramOverCurrentStkyFltCnt	53	Over Current Sticky Fault Count	0	YES
paramOverTempStkyFltCnt	54	Over Temperature Sticky Fault Count	0	YES
paramUnderVoltageStkyFltCnt	55	Under Voltage Sticky Fault Count	0	YES
paramGateDriverStkyFltCnt	56	Get Drive Sticky Fault Count	0	YES
paramCommStkyFltCnt	57	Communications Sticky Fault Count	0 at power on	NO
paramContinuousCurrentLimit	61	Continuous current limit	40 amps	YES
paramPeakCurrentLimit	62	Peak current limit	60 amps	YES

paramPeakCurrentDuration	63	Peak current duration	500 ms	YES
paramCurrentLimitEnable	64	Current limit enable state	Disabled	YES
paramStatusAnalogFrameRate	91	Analog Status Frame Rate	100 ms	NO
paramStatusEncoderFrameRate	92	Quadrature Encoder Status Frame Rate	100 ms	NO
paramStatusGeneralFrameRate	93	General Status Frame Rate	10 ms	NO

Fault Codes

Error	Value	Description
fltOverCurrent	0x0001	Overcurrent fault
fltOverTemp	0x0002	Overtemperature fault
fltUnderVoltage	0x0004	Under voltage fault
fltGateDriver	0x0008	Bridge driver fault (most likely short circuit)
fltComm	0x0010	Communications interface fault

paramLimitSwtFwdNormClosed

Parameters

paramid 1

value

1	Forward limit switch is a normally closed switch
0	Forward limit switch is a normally open switch

Configure the forward limit switch type. The forward limit switch be configured as a normally closed switch by setting the value field to a ‘1’ or a normally open switch by setting the value field to a ‘0’. The DMC60C uses internal (weak) pull-ups to pull the forward limit switch pin to 3.3V. When configured as a normally closed switch the DMC60C will prevent the output from applying a positive voltage to the load when the limit switch opens, causing the DMC60C to detect a logic ‘1’ on the FWDLIM pin. When configured as a normally open switch the DMC60C will prevent the output from applying a positive voltage to the load when the limit switch closes, causing the DMC60C to detect a logic ‘0’ on the FWDLIM pin. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSwtFwdDisabled

Parameters

paramid 2

value

0	Forward limit switch input enabled
1	Forward limit switch input disabled

Configure the forward limit switch enable state. The forward limit switch can be enabled by setting the value field to a ‘0’ or disabled by setting the value field to a ‘1’. When the forward limit switch is disabled the DMC60C will allow the output to apply a positive voltage to the load (when set) regardless of the logic level applied to the FWDLIM pin. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSwtRevNormClosed

Parameters

paramid 3

value

1	Reverse limit switch is a normally closed switch
0	Reverse limit switch is a normally open switch

Configure the reverse limit switch type. The reverse limit switch be configured as a normally closed switch by setting the value field to a ‘1’ or a normally open switch by setting the value field to a ‘0’. The DMC60C uses internal (weak) pull-ups to pull the reverse limit switch pin to 3.3V. When configured as a normally closed switch the DMC60C will prevent the output from applying a negative voltage to the load when the limit switch opens, causing the DMC60C to detect a logic ‘1’ on the REVLIM pin. When configured as a normally open switch the DMC60C will prevent the output from applying a negative voltage to the load when the limit switch closes, causing the DMC60C to detect a logic ‘0’ on the REVLIM pin. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSwtRevDisabled

Parameters

paramid 4

value

0	Reverse limit switch input enabled
1	Reverse limit switch input disabled

Configure the reverse limit switch enable state. The reverse limit switch can be enabled by setting the value field to a ‘0’ or disabled by setting the value field to a ‘1’. When the reverse limit switch is disabled the DMC60C will allow the output to apply a negative voltage to the load (when set) regardless of the logic level applied to the REVLIM pin. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSoftFwdThreshold

Parameters

paramid 5

value 32-bit signed soft forward limit threshold

Configure the soft forward limit threshold. The soft forward limit threshold specifies the maximum position that the encoder can read in the forward direction. The units are native to the encoder that’s connected to the expansion header. The DMC60’s control loop runs every 500us. Each time it executes the current position of the encoder is read and compared to the soft forward limit threshold. If the encoder’s current position is greater than or equal to the specified soft forward limit threshold and the soft forward limit is enabled, then the DMC60’s output will be prevented from applying a positive voltage to the load. Both positive and negative soft limit thresholds are valid. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSoftFwdEnable

Parameters

paramid 6

value

1	Forward soft limit enabled
0	Forward soft limit disabled

Configure the soft forward limit enable state. The soft forward limit can be enabled by setting the value field to a ‘1’ or disabled by setting the value field to a ‘0’. When the soft forward limit is disabled the DMC60’s output will be allowed to apply a positive voltage to the load regardless of the current encoder position and soft forward limit threshold. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSoftRevThreshold

Parameters

paramid 7

value 32-bit signed soft reverse limit threshold

Configure the soft reverse limit threshold. The soft reverse limit threshold specifies the maximum position that the encoder can read in the reverse direction. The units are native to the encoder that's connected to the expansion header. The DMC60's control loop runs every 500us. Each time it executes the current position of the encoder is read and compared to the soft reverse limit threshold. If the encoder's current position is less than or equal to the specified soft reverse limit threshold and the soft reverse limit is enabled, then the DMC60's output will be prevented from applying a negative voltage to the load. Both positive and negative soft limit thresholds are valid. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramLimitSoftRevEnable

Parameters

paramid 8

value

1	Reverse soft limit enabled
0	Reverse soft limit disabled

Configure the soft reverse limit enable state. The soft reverse limit can be enabled by setting the value field to a '1' or disabled by setting the value field to a '0'. When the soft reverse limit is disabled the DMC60's output will be allowed to apply a negative voltage to the load regardless of the current encoder position and soft reverse limit threshold. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramAdcCurrentMultiplier

Parameters

paramid 9

value Signed 16.16 ADC sample to current multiplier

Configure the constant used by the DMC60C to convert ADC measurements into an associated load current in amps. The DMC60C uses a combination of a current sense resistor, bidirectional current sense amplifier with a 50V/V gain, and a 3.3V reference to measure load currents. The current sense amplifier is biased at 1.65V, which means a load current of 0 amps will result in the current sense amplifier outputting 1.65V. At power on the DMC60C performs a calibration procedure to determine the ADC sample value (*smpZeroCurrent*) corresponding to no current flow between the M+ and M- terminals. When current flows from the M+ terminal to the M- terminal the current sense amplifier outputs a voltage between 1.65V and 3.3V, which corresponds to positive current flow. When current flows from the M- terminal to the M+ terminal the current sense amplifier outputs a voltage between 1.65V and 0V, corresponding to negative current flow. The DMC60C uses an internal 12-bit ADC to convert this voltage into digitized value every 500 microseconds. The digitized value is then converted into a signed 16.6 fixed point current measurement (in Amps) using the following formula: $crntLoad = (smpAdc - smpZeroCurrent) \times mplrAdcCurrent$. The multiplier (*mplrAdcCurrent*) that corresponds to a given sense resistance (*resCrntSns*, in ohms) can be calculated using the following formula: $mplrAdcCurrent = (\frac{vref}{4096} \times \frac{1}{resCrntSns \times 50}) \times 65536$. For example, if the sense resistor has a nominal value of 500 µohms then the $mplrAdcCurrent = \left(\frac{3.3}{4096} \times \frac{1}{0.0005 \times 50}\right) \times 65536 = 2112$ or 0x00000840 in hexadecimal. The DMC60C comes pre-programmed with a multiplier that corresponds to the expected sense resistance (approximately 510 µohms) so it should not be necessary to configure the multiplier. However, if current measurements appear to be off then *paramAdcCurrentMultiplier* can be used to adjust the multiplier used. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopPGainSlot0

Parameters

paramid 10

value signed 16.16 closed loop control proportional gain constant

Configure the proportional gain constant used by motor control profile slot 0. This constant is used during closed loop control to calculate a proportional increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopIGainSlot0

Parameters

paramid 11

value signed 16.16 closed loop control integral gain constant

Configure the integral gain constant used by motor control profile slot 0. This constant is used during closed loop control to calculate an integral increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopDGainSlot0

Parameters

paramid 12

value signed 16.16 closed loop control derivative gain constant

Configure the derivative gain constant used by motor control profile slot 0. This constant is used during closed loop control to calculate the derivative increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopIZoneSlot0

Parameters

paramid 13

value 31-bit unsigned integral accumulator limit

Configure the integral accumulator limit used by motor control profile slot 0. The integral accumulator limit, or I-zone, is used to limit how large the integral accumulator can grow during closed loop control. The value sent to the DMC60C is converted to a 32-bit signed integer and used to set the positive and negative bounds of the integral accumulator. If the integral accumulator exceeds these bounds while PID calculations are performed, then the accumulator will be capped to value or -value. This provides a mechanism for combating integral windup. Setting a value of 0 will disable the limit and allow the integral accumulator to grow without bounds. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFGainSlot0

Parameters

paramid 14

value signed 16.16 closed loop control feed forward gain constant

Configure the feed-forward gain constant used by motor control profile slot 0. This constant is used during closed loop control to calculate the number of throttle units to contribute to the duty cycle as the proportion of the setpoint (target Velocity, Position, or Current) independent of the error. For example, if the target current is 20.0 amps and you want to apply 50% throttle for this setpoint then the feed forward gain would be set to $\frac{0.50 \times 32767}{20.0} = 819.175$. Convert this to fixed-point by multiplying by 65536. This results in a value of 0x03332CCC (hex), which is what should be sent to the DMC60C in the *value* field of the PARAMSET packet. The feed-forward term can be excluded from the PID calculations by specifying a value of 0 for the gain. This parameter will be utilized for PID calculations when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopAllowableErrorSlot0

Parameters

paramid	15
value	31-bit unsigned allowable closed loop error

Configure the allowable closed loop error used by motor control profile slot 0. The allowable closed loop error specifies the minimum error required for the PID controller to calculate a non-zero contribution to the output throttle (duty cycle) based on the P, I, and D terms. If the allowable error is set to a non-zero value and the measured error is less than the allowable error then the P, I, and D terms will contribute 0 throttle units to the output throttle calculation and the integral accumulator will be cleared. If the allowable error is set to 0 or the measured error exceeds the allowable error then P, I, and D terms are included in the output throttle calculation. The feed-forward gain constant, or F term, is included in the output throttle calculation regardless of the allowable error setting. This parameter will be utilized for PID calculations when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRampRateSlot0

Parameters

paramid	16
value	31-bit unsigned closed loop ramp rate

Configure the closed loop ramp rate used by motor control profile slot 0. The closed loop ramp rate specifies the maximum number of throttle units the output can change by each time the control loop executes in closed loop control mode (Velocity, Position, or Current). For example, If the closed loop ramp rate is set to 1000 and the PID update function determines that the throttle should be increased by 5000 units then the immediate throttle increase will be limited to 1000 units. If the next PID Update doesn't change the target throttle output value, the throttle will be increased by another 1000 units the next time the control loop executes. This process will continue until the target throttle is reached or a new throttle value is calculated. The control loop executes once every 500 µs. Therefore, specifying a closed loop ramp rate of 16 would result in it taking approximately 1.02 seconds to go from 0% throttle (0) to 100% throttle (32767). Specifying a value of 0 for the closed loop ramp rate disables throttling and allows the output to be immediately set to the target value. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFwdMaxSlot0

Parameters

paramid	17
value	32-bit signed closed loop control maximum forward duty cycle

Configure the closed loop control maximum forward duty cycle used by motor control profile slot 0. The maximum forward duty cycle is the largest positive duty cycle that may be applied to the output when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of 0 to 32767. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRevMaxSlot0

Parameters

paramid 18

value 32-bit signed closed loop control maximum reverse duty cycle

Configure the closed loop control maximum reverse duty cycle used by motor control profile slot 0. The maximum reverse duty cycle is the largest negative duty cycle that may be applied to the output when motor control profile slot 0 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of -32768 to 0. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFwdNominalSlot0

Parameters

paramid 19

value 32-bit signed closed loop control nominal forward duty cycle

Configure the closed loop control nominal forward duty cycle used by motor control profile slot 0. The nominal forward duty cycle is the smallest positive duty cycle that may be applied to the output when the closed loop error exceeds the allowable closed loop error specified for the selected motor profile slot. The closed loop nominal forward duty cycle is only utilized when the control frame specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of 0 to 32767. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRevNominalSlot0

Parameters

paramid 20

value 32-bit signed closed loop control nominal reverse duty cycle

Configure the closed loop control nominal reverse duty cycle used by motor control profile slot 0. The nominal reverse duty cycle is the smallest negative duty cycle that may be applied to the output when the closed loop error exceeds the allowable closed loop error specified for the selected motor profile slot. The closed loop nominal reverse duty cycle is only utilized when the control frame specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of -32768 to 0. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopPGainSlot1

Parameters

paramid 21

value signed 16.16 closed loop control proportional gain constant

Configure the proportional gain constant used by motor control profile slot 1. This constant is used during closed loop control to calculate a proportional increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopIGainSlot1

Parameters

paramid 22

value signed 16.16 closed loop control integral gain constant

Configure the integral gain constant used by motor control profile slot 1. This constant is used during closed loop control to calculate an integral increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopDGainSlot1

Parameters

paramid 23

value signed 16.16 closed loop control derivative gain constant

Configure the derivative gain constant used by motor control profile slot 1. This constant is used during closed loop control to calculate the derivative increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopIZoneSlot1

Parameters

paramid 24

value 31-bit unsigned integral accumulator limit

Configure the integral accumulator limit used by motor control profile slot 1. The integral accumulator limit, or I-zone, is used to limit how large the integral accumulator can grow during closed loop control. The value sent to the DMC60C is converted to a 32-bit signed integer and used to set the positive and negative bounds of the integral accumulator. If the integral accumulator exceeds these bounds while PID calculations are performed, then the accumulator will be capped to value or -value. This provides a mechanism for combating integral windup. Setting a value of 0 will disable the limit and allow the integral accumulator to grow without bounds. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFGainSlot1

Parameters

paramid 25

value signed 16.16 closed loop control feed forward gain constant

Configure the feed-forward gain constant used by motor control profile slot 1. This constant is used during closed loop control to calculate the number of throttle units to contribute to the duty cycle as the proportion of the setpoint (target Velocity, Position, or Current) independent of the error. For example, if the target current is 20.0 amps and you want to apply 50% throttle for this setpoint then the feed forward gain would be set to $\frac{0.50 \times 32767}{20.0} = 819.175$. Convert this to fixed-point by multiplying by 65536. This results in a value of 0x03332CCC (hex), which is what should be sent to the DMC60C in the *value* field of the PARAMSET packet. The feed-forward term can be excluded from the PID calculations by specifying a value of 0 for the gain. This parameter will be utilized for PID calculations when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopAllowableErrorSlot1

Parameters

paramid 26

value 31-bit unsigned allowable closed loop error

Configure the allowable closed loop error used by motor control profile slot 1. The allowable closed loop error specifies the minimum error required for the PID controller to calculate a non-zero contribution to the output throttle (duty cycle) based on the P, I, and D terms. If the allowable error is set to a non-zero value and the measured error is less than the allowable error then the P, I, and D terms will contribute 0 throttle units to the output throttle calculation and the integral accumulator will be cleared. If the allowable error is set to 0 or the measured error exceeds the allowable error then P, I, and D terms are included in the output throttle calculation. The feed-forward gain constant, or F term, is included in the output throttle calculation regardless of the allowable error setting. This parameter will be utilized for PID calculations when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRampRateSlot1

Parameters

paramid 27

value 31-bit unsigned closed loop ramp rate

Configure the closed loop ramp rate used by motor control profile slot 1. The closed loop ramp rate specifies the maximum number of throttle units the output can change by each time the control loop executes in closed loop control mode (Velocity, Position, or Current). For example, If the closed loop ramp rate is set to 1000 and the PID update function determines that the throttle should be increased by 5000 units then the immediate throttle increase will be limited to 1000 units. If the next PID Update doesn't change the target throttle output value, the throttle will be increased by another 1000 units the next time the control loop executes. This process will continue until the target throttle is reached or a new throttle value is calculated. The control loop executes once every 500 µs. Therefore, specifying a closed loop ramp rate of 16 would result in it taking approximately 1.02 seconds to go from 0% throttle (0) to 100% throttle (32767). Specifying a value of 0 for the closed loop ramp rate disables throttling and allows the output to be immediately set to the target value. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFwdMaxSlot1

Parameters

paramid 28

value 32-bit signed closed loop control maximum forward duty cycle

Configure the closed loop control maximum forward duty cycle used by motor control profile slot 1. The maximum forward duty cycle is the largest positive duty cycle that may be applied to the output when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of 0 to 32767. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRevMaxSlot1

Parameters

paramid 29

value 32-bit signed closed loop control maximum reverse duty cycle

Configure the closed loop control maximum reverse duty cycle used by motor control profile slot 1. The maximum reverse duty cycle is the largest negative duty cycle that may be applied to the output when motor control profile slot 1 is specified in a control frame that specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of -32768 to 0. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopFwdNominalSlot1

Parameters

paramid 30

value 32-bit signed closed loop control nominal forward duty cycle

Configure the closed loop control nominal forward duty cycle used by motor control profile slot 1. The nominal forward duty cycle is the smallest positive duty cycle that may be applied to the output when the closed loop error exceeds the allowable closed loop error specified for the selected motor profile slot. The closed loop nominal forward duty cycle is only utilized when the control frame specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of 0 to 32767. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramClosedLoopRevNominalSlot1

Parameters

paramid 31

value 32-bit signed closed loop control nominal reverse duty cycle

Configure the closed loop control nominal reverse duty cycle used by motor control profile slot 1. The nominal reverse duty cycle is the smallest negative duty cycle that may be applied to the output when the closed loop error exceeds the allowable closed loop error specified for the selected motor profile slot. The closed loop nominal reverse duty cycle is only utilized when the control frame specifies one of the closed loop control modes (Velocity, Position, or Current). The value specified for this parameter should be restricted to be within the range of -32768 to 0. This parameter is stored in nonvolatile memory and is preserved across power cycles.

paramCurrentLimitPGain

Parameters

paramid 32

value signed 16.16 proportional gain constant used for current limiting

Configure the proportional gain constant used by motor controller while performing current limiting. This constant is used to calculate a proportional increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when current limiting is active. This parameter is stored in nonvolatile memory and is preserved across power cycles. The default value should be sufficient for most applications and should be tested before any adjustments are made.

paramCurrentLimitIGain

Parameters

paramid 33

value signed 16.16 integral gain constant used for current limiting

Configure the integral gain constant used by motor controller while performing current limiting. This constant is used during closed loop control to calculate an integral increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when current limiting is active. This parameter is stored in nonvolatile memory and is preserved across power cycles. The default value should be sufficient for most applications and should be tested before any adjustments are made.

paramCurrentLimitDGain

Parameters

paramid 34

value signed 16.16 derivative gain constant used for current limiting

Configure the derivative gain constant used by motor controller while performing current limiting. This constant is used during closed loop control to calculate the derivative increase or decrease in the throttle (duty cycle) due to the measured closed loop error. This parameter will be utilized for PID calculations when current limiting is active. This parameter is stored in nonvolatile memory and is preserved across power cycles. The default value should be sufficient for most applications and should be tested before any adjustments are made.

paramCurrentLimitIZone

Parameters

paramid 35

value 31-bit unsigned integral accumulator limit used for current limiting

Configure the integral accumulator limit used by motor controller while performing current limiting. The integral accumulator limit, or I-zone, is used to limit how large the integral accumulator used for current limiting can grow when current limiting is active. The value sent to the DMC60C is converted to a 32-bit signed integer and used to set the positive and negative bounds of the integral accumulator. If the integral accumulator exceeds these bounds while PID calculations are performed, then the accumulator will be capped to value or -value. This provides a mechanism for combating integral windup. Setting a value of 0 will disable the limit and allow the integral accumulator to grow without bounds. This parameter is stored in nonvolatile memory and is preserved across power cycles. The default value should be sufficient for most applications and should be tested before any adjustments are made.

paramCurrentLimitFGain

Parameters

paramid 36

value signed 16.16 feed forward gain constant used for current limiting

Configure the feed-forward gain constant used by motor controller while performing current limiting. This constant is used during closed loop control to calculate the number of throttle units to contribute to the duty cycle as the proportion of the setpoint independent of the error. For example, if the target current is 20.0 amps and you want to apply 50% throttle for this setpoint then the feed forward gain would be set to $\frac{0.50\times 32767}{20.0} = 819.175$. Convert this to fixed-point by multiplying by 65536. This results in a value of 0x03332CCC (hex), which is what should be sent to the DMC60C in the *value* field of the PARAMSET packet. The feed-forward term can be excluded from the PID calculations by specifying a value of 0 for the gain. This parameter will be utilized for PID calculations when current limiting is active. This parameter is stored in nonvolatile memory and is preserved across power cycles. The default value should be sufficient for most applications and should be tested before any adjustments are made.

paramEncoderPosition

Parameters

paramid 41

value 32-bit signed encoder position

Configure the encoder position. The encoder position is maintained by the QEI module of the DMC60's MCU and is continuously updated by the pulse train applied to QEA and QEB inputs of the expansion header. At power on the encoder's position is initialized to zero. This may not correspond with the zero point that's defined in the end user application, and as such, it may be necessary to set the encoder to a specific position or reset it to 0 after performing a homing sequence. The encoder's position is used for closed loop position control and for determining whether the forward soft limit or reverse soft limit have been hit. The encoder position should only be configured while the DMC60's output is disabled.

paramClearPositionOnIndex

Parameters

paramid 42

value

1	Encoder position count is cleared by the active edge on the Index pin
0	Encoder position count is unaffected by the Index pin

Configure the index pin to clear the encoder position count. When a '1' is specified in the value field the detection of the configured active edge (rising or falling) on the Index pin will cause the encoder position count to be cleared. Specifying a '0' in the value field causes the DMC60C to ignore the state of the index pin. The Index pin features an internal pull-up. This makes it possible to connect a normally closed or normally open switch between the index pin and ground. This parameter is stored in volatile memory and is not preserved across power cycles.

paramClearPositionOnFwdLimit

Parameters

paramid 43

value

1	Encoder position count is cleared by the forward limit switch
0	Encoder position count is unaffected by the forward limit switch

Configure the forward limit switch to clear the encoder position count. When a '1' is specified in the value field, the encoder position count will be automatically cleared when the forward limit switch is active. The position count will continue to be cleared for as long as the forward limit switch remains active. The active state of the forward limit switch can be configured as normally open or normally closed. The forward limit switch does not have to be enabled for the encoder position to be cleared. When a '0' in the value field the encoder position is unaffected by the state of the forward limit switch. This parameter is stored in volatile memory and is not preserved across power cycles.

paramClearPositionOnRevLimit

Parameters

paramid 44

value

1	Encoder position count is cleared by the reverse limit switch
0	Encoder position count is unaffected by the reverse limit switch

Configure the reverse limit switch to clear the encoder position count. When a '1' is specified in the value field, the encoder position count will be automatically cleared when the reverse limit switch is active. The position count will continue to be cleared for as long as the reverse limit switch remains active. The active state of the reverse limit switch can be configured as normally open or normally closed. The reverse limit switch does not have to be enabled for the encoder position to be cleared. When a '0' in the value field the encoder position is unaffected by the state of the reverse limit switch. This parameter is stored in volatile memory and is not preserved across power cycles.

paramIndexActiveEdge

Parameters

paramid 45

value

1	Index pin detects rising edges
0	Index pin detects falling edges

Configure the index pin to detect rising or falling edges. When a '1' is specified in the value field the index pin will detect an index event whenever a rising edge occurs. When a '0' is specified in the value field the index pin will detect an index event whenever a falling edge occurs. This parameter is stored in volatile memory and is not preserved across power cycles.

paramActiveFaults

Parameters

paramid 51

value Not applicable

Get a field set (bit field) containing the active faults. A value of '1' in the corresponding bit position indicates that the associated fault is present. A value of '0' indicates that the fault isn't currently present. When the DMC60C detects a fault it disables its output, sets a 3 second countdown timer, and enters the fault state. If no faults are present after the countdown timer expires then the DMC60C will exit the fault state and return to running or waiting for link mode (PWM input or CAN input). If another fault occurs before the countdown timer expires, or the fault is still present after the timer expires, then timer is reset to 3 seconds and the DMC60C remains in the fault state.

paramStickyFaults

Parameters

paramid 52

value Non-zero to clear the sticky faults after reading them, 0 to keep them in tact

Get a field set (bit field) containing the sticky faults. A value of '1' in the corresponding bit position indicates that the associated fault has occurred at some point, even if it's not currently present. A value of '0' indicates that the fault has not occurred since the last time the sticky faults were cleared. When the DMC60C detects a fault, it sets the bit associated with that fault in a variable that keeps track of the sticky faults. Additionally, it increments a count variable that keeps track of the number of times the fault has occurred. This allows the Robot Controller to detect intermittent faults conditions that may have occurred. The sticky fault flags are stored in nonvolatile memory and are preserved across power cycles.

paramOverCurrentStkyFltCnt

Parameters

paramid 53

value Non-zero to reset the over current sticky fault count after reading it, 0 to maintain the existing count value

Get the Over Current Sticky Fault Count. The Over Current Sticky Fault Count is the number of times that a new overcurrent fault has occurred. Specifying a non-zero value in the *value* field of the *PARAMSET* packet will clear the count after it has been read into the *PARAMRESP* packet. The overcurrent sticky fault count is stored in nonvolatile memory and is preserved across power cycles.

paramOverTempStkyFltCnt

Parameters

paramid 54

value Non-zero to reset the over temperature sticky fault count after reading it, 0 to maintain the existing count value

Get the Over Temperature Sticky Fault Count. The Over Temperature Sticky Fault Count is the number of times that a new over temperature fault has occurred. Specifying a non-zero value in the *value* field of the *PARAMSET* packet will clear the count after it has been read into the *PARAMRESP* packet. The over temperature fault count is stored in nonvolatile memory and is preserved across power cycles.

paramUnderVoltageStkyFltCnt

Parameters

paramid 55

value Non-zero to reset the under voltage sticky fault count after reading it, 0 to maintain the existing count value

Get the Under Voltage Sticky Fault Count. The Under Voltage Sticky Fault Count is the number of times that a new under voltage fault has occurred. Specifying a non-zero value in the *value* field of the *PARAMSET* packet will clear the count after it has been read into the *PARAMRESP* packet. An under voltage fault occurs when the input voltage is below 5.75 volts for 5 consecutive seconds. The under voltage fault count is stored in nonvolatile memory and is preserved across power cycles.

paramGateDriverStkyFltCnt

Parameters

paramid 56

value Non-zero to reset the gate driver sticky fault count after reading it, 0 to maintain the existing count value

Get the Gate Driver Sticky Fault Count. The Gate Driver Sticky Fault Count is the number of times that a new bridge driver fault has occurred. Specifying a non-zero value in the *value* field of the *PARAMSET* packet will clear the count after it has been read into the *PARAMRESP* packet. Gate driver faults typically indicate that a short-circuit has occurred. Therefore, the user should check to see if the M+ or M- leads are shorted to the chassis, each other, or the power supply. The gate driver fault count is stored in nonvolatile memory and is preserved across power cycles.

paramCommStkyFltCnt

Parameters

paramid 57

value Non-zero to reset the communications sticky fault count after reading it, 0 to maintain the existing count value

Get the Communications Sticky Fault Count. The Communications Sticky Fault Count is the number of times that the connection to the CAN bus has been lost since the last power cycle. Specifying a non-zero value in the *value* field of the *PARAMSET* packet will clear the count after it has been read into the *PARAMRESP* packet.

paramContinuousCurrentLimit

Parameters

paramid	61
value	signed 16.16 continuous current limit (in amps)

Configure the Continuous Current Limit. The DMC60C continuously measures the load current. If load current exceeds the Peak Current Limit for longer than the Peak Current Duration and current limiting is enabled, then the load current will be limited to the value specified by the Continuous Current Limit. If the Continuous Current Limit is set to a value that's greater than or equal to the Peak Current Limit and current limiting is enabled, then the DMC60C will limit begin limiting the load current immediately after the first time that it detects that the Continuous Current Limit has been exceeded.

paramPeakCurrentLimit

Parameters

paramid	62
value	signed 16.16 peak current limit (in amps)

Configure the Peak Current Limit. The DMC60C continuously measures the load current. If load current exceeds the Peak Current Limit for longer than the Peak Current Duration and current limiting is enabled, then the load current will be limited to the value specified by the Continuous Current Limit. If the Peak Current Duration is set to 0 and current limiting is enabled, then the DMC60C will begin applying the Continuous Current Limit immediately after the first time that it detects that the Peak Current Limit has been exceeded. If the Peak Current Limit is set to a value that's smaller than the Continuous Current Limit and current limiting is enabled, then the DMC60C will begin applying the Continuous Current Limit immediately after it detects that the Continuous Current Limit has been exceeded.

paramPeakCurrentDuration

Parameters

paramid	63
value	16-bit unsigned peak current duration (in milliseconds)

Configure the Peak Current Duration. The DMC60C continuously measures the load current. If load current exceeds the Peak Current Limit for longer than the Peak Current Duration and current limiting is enabled, then the load current will be limited to the value specified by the Continuous Current Limit. If the Peak Current Duration is set to 0 and current limiting is enabled, then the DMC60C will begin applying the Continuous Current Limit immediately after the first time that it detects that the Peak Current Limit has been exceeded. If the Peak Current Limit is set to a value that's smaller than the Continuous Current Limit and current limiting is enabled, then the DMC60C will begin applying the Continuous Current Limit immediately after it detects that the Continuous Current Limit has been exceeded.

paramCurrentLimitEnable

Parameters

paramid 64

value

1	Current Limit is enabled
0	Current Limit is disabled

Configure the Current Limit enable state. The Current Limit can be enabled by setting the value field to a '1' or disabled by setting the value field to a '0'. When the Current Limit is disabled the DMC60C will not limit the load current regardless of the values specified for the Continuous Current Limit, Peak Current Limit, and Peak Current Duration.

paramStatusAnalogFrameRate

Parameters

paramid	91
value	32-bit unsigned status message frame rate in milliseconds

Configure the rate at which the DMC60C transmits Analog Input, Current, Temperature, and Battery Voltage Status Frames. These status frames are formatted as *STSANALOG* packets and are described in the Periodic Status Messages section. The frame rate can be set to any value between 1 millisecond and 30000 milliseconds. Power cycling the DMC60C will result in the device reverting to the default frame rate, which is 100 milliseconds.

paramStatusEncoderFrameRate

Parameters

paramid 92

value 32-bit unsigned status message frame rate in milliseconds

Configure the rate at which the DMC60C transmits Quadrature Encoder Status Frames. These status frames are formatted as *STSENCODER* packets and are described in the Periodic Status Messages section. The frame rate can be set to any value between 1 millisecond and 30000 milliseconds. Power cycling the DMC60C will result in the device reverting to the default frame rate, which is 100 milliseconds.

paramStatusGeneralFrameRate

Parameters

paramid 93

value 32-bit unsigned status message frame rate in milliseconds

Configure the rate at which the DMC60C transmits General Status Frames. These status frames are formatted as *STSGENERAL* packets and are described in the Periodic Status Messages section. The frame rate can be set to any value between 1 millisecond and 50 milliseconds. Power cycling the DMC60C will result in the device reverting to the default frame rate, which is 10 milliseconds.

Periodic Status Frames

Overview

Once the DMC60C has detected the presence of the CAN bus it will begin transmitting periodic status frames. Periodic status frames provide regular feedback to the Robot Controller (or host) and may be useful in implementing certain types of control applications. Additionally, they provide information that may be useful for debugging closed loop control configuration parameters.

The rate at which periodic status frames are broadcast is specific to each type of status frame and can be adjusted setting the appropriate configuration parameter. The DMC60C will contain to broadcast status frames until it detects the loss of the CAN bus. This occurs when the DMC60C has transmitted too many frames that haven't received an acknowledgement (indicating loss of Robot Controller) or too many consecutive frame errors occur.

Message Identifiers and Data Structures

Periodic Status Frame Message Identifiers

Message Identifier	Value	Default Period
msgidStsGeneral	0x02061400	10 milliseconds
msgidStsEncoder	0x02061480	100 milliseconds
msgidStsAnalog	0x020614C0	100 milliseconds

Note: All message identifiers transmitted as part of a period status frame should include a Device Number in the lower 6 bits of the extended identifier.

STSGENERAL Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
errCLoopH	errCLoopM	errCLoopL	fs2	fs1	fs0	dtcApplied	

dtcApplied Signed 16-bit integer corresponding to the output duty cycle currently applied to the H-Bridge

fs0

Bit	3	2	1	0
	fFwdLimitNormalClosed	fFwdLimitDisabled	fFwdLimitHit	fFwdLimitPin

fFwdLimitPin 1 when forward limit pin is high, 0 otherwise
fFwdLimitHit 1 when forward limit is active, 0 otherwise
fFwdLimitDisabled 1 when forward limit is disabled, 0 when forward limit is enabled
fFwdLimitNormalClosed 1 when forward limit switch is normally closed, 0 when normally open

Bit	7	6	5	4
	fRevLimitNormalClosed	fRevLimitDisabled	fRevLimitHit	fRevLimitPin

fRevLimitPin 1 when the reverse limit pin is high, 0 otherwise
fRevLimitHit 1 when the reverse limit is active, 0 otherwise
fRevLimitDisabled 1 when the reverse limit is disabled, 0 when reverse limit is enabled
fRevLimitNormalClosed 1 when reverse limit switch is normally closed, 0 when normally open

fs1

Bit	3	2	1	0
	fSoftFwdLimitHit	fRevLimitDisableOvrd	fFwdLimitDisableOvrd	fFwdRevLimitOverride

fFwdRevLimitOverride 1 when limit switch enable state is being overridden by the control frame, 0 otherwise
fFwdLimitDisableOvrd 1 when forward limit switch is disabled by the control frame, 0 when forward limit switch is enabled by the control frame
fRevLimitDisableOvrd 1 when reverse limit switch is disabled by the control frame, 0 when reverse limit switch is enabled by the control frame
fSoftFwdLimitHit 1 when forward soft limit is active, 0 otherwise

Bit	7	6	5	4
	fCurrentLimitActive	fSoftRevLimitEnabled	fSoftRevLimitHit	fSoftFwdLimitEnabled

fSoftFwdLimitEnabled 1 when soft forward limit is enabled, 0 when disabled
fSoftRevLimitHit 1 when reverse limit is active, 0 otherwise
fSoftRevLimitEnabled 1 when soft reverse limit is enabled, 0 when disabled
fCurrentLimitActive 1 when the current limit is being enforced, 0 otherwise

fs2

Bit	7	6	5	4	3	2	1	0
	fDivErrBy256	modeSelect				fGateDriverFault	fUnderVoltageFault	fOverTempFault

fOverTempFault 1 when over temperature fault is active, 0 otherwise
fUnderVoltageFault 1 when under voltage fault is active, 0 otherwise
fGateDriverFault 1 when bridge driver fault is active, 0 otherwise

modeSelect Control Mode
0: modeVoltage - Open Loop Voltage Control (Duty Cycle)
1: modeVelocity - Closed Loop Velocity Control
2: modePosition - Closed Loop Position Control
3: modeCurrent - Closed Loop Current Control
4: modeVComp - Voltage Compensation Mode
5: modeFollower - Slave Follower Mode
15: modeNoDrive - No Drive (output disabled)
Note: all other values reserved for future use.

fDivErrBy256 1 if the returned closed loop error has been divided by 256, 0 otherwise

errCLoopL Low byte of the Closed Loop Error. Meaning is specific the control mode specified in modeSelect
errCLoopM Middle byte of the Closed Loop Error. Meaning is specific the control mode specified in modeSelect
errCLoopH High byte of the Closed Loop Error. Meaning is specific the control mode specified in modeSelect

STSENCODER Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
flgsEnc	rsv1	rsv0	velcntL	velcntH	poscntL	poscntM	poscntH

poscntH High byte of the encoder's current position count. Units are native to the encoder being used.
poscntM Middle byte of the encoder's current position count. Units are native to the encoder being used.
poscntL Low byte of the encoder's current position count. Units are native to the encoder being used.
velcntH High byte of the velocity count in native units per 100 milliseconds.
velcntL Low byte of the velocity count in native units per 100 milliseconds.
rsv0 Reserved for future use.
rsv1 Reserved for future use.

flgsEnc

Bit	7	6	5	4	3	2	1	0
	rsv2			fQEIdxPin	fQEBPin	fQEAPin	fDivVelBy4	fDivPosBy8

fDivPosBy8 1 if the returned position count has been divided by 8.
fDivVelBy4 1 if the returned velocity count has been divided by 4.
fQEAPin 1 when quadrature encoder A pin is high, 0 otherwise.
fQEBPin 1 when quadrature encoder B pin is high, 0 otherwise.
fQEIdxPin 1 when quadrature encoder Index pin is high, 0 otherwise.
rsv2 Reserved for future use.

STSANALOG Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
vltgVbus		tmpAmbient		crntOutput		vltgAnalogIn	

vltgAnalogIn Signed 8.8 input voltage (in volts) corresponding to the voltage on the AIN1 pin.
crntOutput Signed 8.8 load current (in amps) corresponding to the H-Bridge output.
tmpAmbient Signed 8.8 ambient temperature (in degrees C) corresponding to the internal case temperature.
vltgVbus Signed 8.8 input voltage (in volts) corresponding to the motor controller's bus (battery) voltage.

Quadrature Encoder Status Frames

The DMC60's internal hardware features a quadrature decoder module that monitors each edge transition on the channel A and B signals of an attached encoder. Each time an edge transition is detected, the internal position and velocity count registers are incremented or decremented. An encoder that specifies 1024 cycles per revolution will report a position count value of 4096 for each revolution that occurs in the forward direction.

Quadrature encoder status frames are encapsulated in a *STSENCODER* packet and transmitted with *msgidStsEncoder*. These frames include the position and velocity count measured by a quadrature encoder that may be attached to expansion header of the DMC60. The position count is returned as a signed 24-bit value through the *poscntH*, *poscntM*, and *poscntL* fields of the *STSENCODER* packet. The Robot Controller (or host) should multiply the received position count value by 8 when the *fDivPosBy8* field is set to a '1'. In effect this makes the position count a signed 27-bit value and limits the measured position to be between -67108864 and 67100863.

Velocity is returned as a signed 16-bit value through the *velcntH* and *velcntL* fields of the *STSENCODER* packet. The Robot Controller (or host) should multiply the received velocity count value by 4 when the *fDivVelBy4* field is set to a '1'. In effect this makes the velocity count a signed 18-bit value and limits measured velocity count to be between -131072 and 131071. The maximum velocity (in RPM) that can be measured by the DMC60C depends on the resolution of the encoder being used. While higher resolution encoders provide better accuracy, they also reduce the maximum velocity that can be measured. For example, an encoder that specifies 1024 cycles per revolution (4096 counts per revolution) will be limited to measuring velocities between -19200 RPM and 19199.85352 RPM. If a wider measurement range is required, then a lower resolution encoder must be used.

Velocity is returned as a signed value that indicates the number of native counts (*velcnt*) that have occurred over a 100ms period. This value can be converted into revolutions per minute (RPM) using the following formulate: $RPM = \frac{velcnt \times 600}{CPR}$. Please note that CPR is the number of counts per revolution (4x the number of cycles per revolution). For example, if an encoder specifies 20 cycles per revolution then the formula used to calculate RPM is $RPM = \frac{velcnt \times 600}{80}$.

Analog Input, Current, Temperature, and Battery Voltage Status Frame

The DMC60C features onboard circuitry for monitoring an external voltage applied to the AIN1 pin of the expansion header, the load current of the H-Bridge, the ambient temperature inside of the case, and the bus voltage (battery input voltage). The firmware uses the on-chip analog to digital converter (ADC) to measure the voltage output by each of monitoring circuits every 500 µs. The digitized values are then scaled and converted to the appropriate format for monitoring and performing closed loop control. This information is encapsulated in a *STSANALOG* packet and transmitted with *msgidStsAnalog*.

Each *STSANALOG* packet contains four fields: *vltgAnalogIn*, *crntOutput*, *tmpAmbient*, and *vltgVbus*. The *vltgAnalogIn* field is a signed 8.8 fixed point value that contains the voltage (in volts) measured on the AIN1 pin of the expansion header. The AIN1 pin is capable of measuring voltages between 0 and 3.3 volts. The *crntOutput* field is a signed 8.8 fixed point value that contains the current (in amps) that's being consumed by a load attached to the output of the H-Bridge. The *tmpAmbient* field is a signed 8.8 fixed point value that contains the ambient temperature (in degrees C) measured inside the case by the onboard temperature sensor. The *vltgVbus* field is a signed 8.8 fixed point value that contains the voltage (in volts) measured for the battery input voltage.

Output Control Protocol

Overview

The DMC60C supports a variety of open loop and closed loop control modes, which may be used to drive a motor attached to the M+ and M- wires. The following modes are presently supported: Voltage Mode (open loop duty cycle), Closed Loop Velocity Mode, Closed Loop Position Mode, Closed Loop Current Mode, Voltage Compensation Mode, and Slave Follower Mode. The use of the Closed Loop Position or Closed Loop Velocity modes requires an external quadrature encoder to be attached to the DMC60's expansion header.

The desired control mode and setpoint (target duty cycle, velocity, position, current, or voltage) are specified by the Robot Controller (or host) as part of a *CANCTRL0* packet that's transmitted with the *msgidControl0* identifier. Each time the DMC60C receives a *CANCTRL0* packet it makes any required adjustments to its output and then restarts an internal heartbeat timer, which is used to implement a 104 millisecond timeout. If the heartbeat timer expires then the DMC60's output is disabled, and it enters a halted state, which will cause the LEDs to display orange, alternating side to side. The DMC60C will then remain in the halted state until a new control frame specifying a non-zero setpoint is received or until the loss of the CAN bus is detected. Therefore, it is important for the Robot Controller to send control frames to the DMC60C on a regular basis. Specifying *modeNoDrive* in the *modeSelect* field will also place the DMC60C in the halted state.

The target setpoint is specified through the *trgtH*, *trgtM*, and *trgtL* fields of the *CANCTRL0* packet. The units specified for the *trgt* field are specific to the control mode (*modeVoltage*, *modeVelocity*, *modePosition*, *modeCurrent*, *modeVComp, modeFollower*) that's specified in the *modeSelect* field. When Voltage Mode (*modeVoltage*) is specified in the *modeSelect* field *trgtM* and *trgtL* specify the 16-bit signed duty cycle used to drive the H-Bridge. If a value of 0 is specified in the *vltgRampSet* field, then the output throttle will be immediately set to the target duty cycle the next time the control loop executes. If a non-zero value is specified in the *vltgRampSet* field, then the number of throttle units that the output can change by each time the control loop executes will be limited to the value that was specified. For example, if the output throttle is currently set to 5000, the *trgt* field specifies 10000, and the *vltgRampSet* field specifies 2500, then the control loop will need to execute twice before the output throttle is set to a target duty cycle of 10000. The control loop executes every 500 µs. In this example it may take up to 1 millisecond for the output throttle to be set to the target duty cycle.

When Closed Loop Velocity Mode (*modeVelocity*) is specified in the *modeSelect* field *trgtH*, *trgtM*, and *trgtL* specify a 24-bit signed value that corresponds to the number of native counts (encoder ticks) required in a 100ms period to achieve the desired velocity. The number of native counts (*velcnt*) required to achieve a specific RPM can be calculated using the following formula: $velcnt = \frac{RPM \times CPR}{600}$. Please note that *CPR* is the number of counts per revolution (4x the number of cycles per revolution) specified for the quadrature encoder attached to the DMC60's expansion header. Each time the control loop executes in Closed Loop Velocity Mode it calculates the error between measured velocity and the target velocity, uses the constants associated with the motor control profile slot specified by the *pidsltSelect* field to calculate a target throttle, and then adjusts the H-Bridge's output throttle.

When Closed Loop Position Mode (*modePosition*) is specified in the *modeSelect* field *trgtH*, *trgtM*, and *trgtL* specify a 24-bit signed value that corresponds to the number of position counts (encoder ticks) required to achieve the desired position. Each time the control loop executes in Closed Loop Position Mode it calculates the error between measured position and the target position, uses the constants associated with the motor control profile slot specified by the *pidsltSelect* field to calculate a target throttle, and then adjusts the H-Bridge's output throttle.

When Closed Loop Current Mode (*modeCurrent*) is specified in the *modeSelect* field *trgtH*, *trgtM*, and *trgtL* specify a signed 8.16 fixed point value that corresponds to the desired load current in amps. Each time the control loop executes in Closed Loop Current Mode it calculates the error between the measured load current and the target load current, uses the constants associated with the motor control profile slot specified by the *pidsltSelect* field to calculate a target throttle, and then adjusts the H-Bridge's output throttle.

When Voltage Compensation Mode (*modeVComp*) is specified in the *modeSelect* field *trgtH*, *trgtM*, and *trgtL* specify a signed 8.16 fixed point value that corresponds to the desired output voltage in volts. Each time the control loop executes in Voltage Compensation Mode the input voltage (bus voltage) is measured and used to compute the target duty cycle required to achieve the specified target voltage. If the specified output voltage exceeds the input voltage, then the computed target duty cycle is limited to 100% (32767 or -32768). If a value of 0 is specified in the *vltgRampSet* field, then the output throttle will be immediately set to the target duty cycle. If a non-zero value is specified in the *vltgRampSet* field, then the number of throttle units that the output can change by each time the control loop executes will be limited to the value that was specified. For example, if the output throttle is currently set to 5000, the computed target duty cycle is 10000, and the *vltgRampSet* field specifies 2500, then output throttle will be set to 7500. If the target duty cycle remains the same, then the output throttle will be set to 10000 the next time the control loop executes (500 µs later).

When Slave Follower Mode (*modeFollower*) is specified in the *modeSelect* field the *trgtH* and *trgtM* fields are ignored and the *trgtL* field is used to specify the device number of the DMC60C (master) to be followed. In Slave Follower Mode the DMC60C behaves similar to Voltage Mode (*modeVoltage*). The primary difference between the two modes is that the output throttle is set to replicate that of the master, which the master broadcasts through periodic *STSGENERAL* packets with *msgidStsGeneral*. If the DMC60C does not receive a valid *STSGENERAL* packet from the master within 104ms then a timeout will occur, and the output throttle will be set to zero.

The *fOverrideBC* and fBrakeCoast fields of the *CANCTRL0* packet allow the Robot Controller to override the existing Brake / Coast Mode setting. Specifying a '1' in the *fOverrideBC* field will result in the DMC60C either Braking (*fBrakeCoast* = '1') or Coasting (*fBrakeCoast* = '0') when the neutral throttle is applied regardless of the previous Brake / Coast setting. The Brake / Coast LED is updated accordingly to reflect the active setting. If the control frame specifies a '1' for *fOverrideBC* and then later specifies '0', the DMC60C reverts to the previous Brake / Coast setting.

The *fRevFeedbackSensor* field of the *CANCTRL0* packet allows the Robot Controller to instruct the DMC60C to reverse direction of the feedback sensor (quadrature encoder). If a '1' is specified for *fRevFeedbackSensor* and rotation in the clockwise direction previously resulted in a positive position/velocity count, then rotation in the clockwise direction will now result in a negative position/velocity count. This alleviates the need to physically swap the QEA and QEB signals when the direction of the quadrature encoder does not match that of the motor. Please note that the *fRevFeedbackSensor* field is ignored while operating in Slave Follower Mode.

The *fRevMotor* field of the *CANCTRL0* packet allows the Robot Controller to instruct the DMC60C reverse the direction of its output. If a '1' is specified for *fRevMotor* and positive target value previously resulted in clockwise rotation, then a positive target value will now result in counter-clockwise rotation. This alleviates the need to physically swap the M+ and M- connections to the motor, which would typically be required when mounting motors opposite of one another on a drivetrain.

The *fsLimitOverride* field of the *CANCTRL0* packet contains the *fEnableLimitOverride*, *fDisableFwdLimit*, and *fDisableRevLimit* flags. When the *fEnableLimitOverride* flag is set to a '1' the *fDisableFwdLimit* and *fDisableRevLimit* flags specify whether the forward and reverse limit switch inputs are enabled. When the *fEnableLimitOverride* flag is set to a '0' the forward and reverse limit switch inputs behave as configured through their applicable configuration parameters.

Message Identifiers and Data Structures

Control 0 Message Identifier

Message Identifier	Value
msgidControl0	0x02060000
Note: All message identifiers transmitted as part of a control frame should include a Device Number in the lower 6 bits of the extended identifier.	

CANCTRL0 Data Structure

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
vltgRampSet		trgtM	trgtL	trgtH	rsv2	fs1	fs0

fs0

Bit	7	6	5	4	3	2	1	0
	fRevFeedbackSensor	pidsltSelect	fBrakeCoast	fOverrideBC	modeSelect			

modeSelect

Control Mode
- 0: modeVoltage - Open Loop Voltage Control (Duty Cycle)
- 1: modeVelocity - Closed Loop Velocity Control
- 2: modePosition - Closed Loop Position Control
- 3: modeCurrent - Closed Loop Current Control
- 4: modeVComp - Voltage Compensation Mode
- 5: modeFollower - Slave Follower Mode
- 15: modeNoDrive - No Drive (output disabled)

Note: all other values reserved for future use.

fOverrideBC Enable Brake / Coast Mode Override
- 1: Override brake/coast mode setting with the one specified by fBrakeCoast
- 0: Use existing brake/coast mode setting and ignore fBrakeCoast

fBrakeCoast Brake / Coast Override Setting
- 1: Brake when neutral throttle is applied
- 0: Coast when neutral throttle is applied

pidsltSelect Motor control profile slot used for closed loop control modes
- 1: Slot 1 selected during closed loop control mode
- 0: Slot 0 selected during closed loop control mode

fRevFeedbackSensor Reverse the direction of the feedback sensor
- 1: Reverse feedback sensor direction for closed loop velocity/position mode
- 0: Use normal feedback sensor direction for closed loop velocity/position mode

fs1

Bit	7	6	5	4	3	2	1	0
	rsv1				fsLimitOverride			fRevMotor

fRevMotor Reverse Motor Direction
- 1: Motor is driven in the opposite direction of that specified by the setpoint
- 0: Motor is driven in the direction specified by the setpoint

fsLimitOverride

Bit	3	2	1
	fEnableLimitOverride	fDisableRevLimit	fDisableFwdLimit

fDisableFwdLimit Enable / Disable Forward Limit Switch
- 1: Disable forward limit switch
- 0: Enable forward limit switch

fDisableRevLimit Enable / Disable Reverse Limit Switch
- 1: Disable reverse limit switch
- 0: Enable forward limit switch

fEnableLimitOverride Enable / Disable Limit Switch Override
- 1: Enable limit switch override
- 0: Disable limit switch override

rsv1 Reserved for future use

rsv2 Reserved for future use
trgtH High byte of target setpoint. Meaning is specific to the control mode specified in modeSelect
trgtL Low byte of target setpoint. Meaning is specific to the control mode specified in modeSelect
trgtM Middle byte of target setpoint. Meaning is specific to the control mode specified in modeSelect
vltgRampSet 16-bit unsigned throttle ramp rate