

## **DESI Bootloader Documentation V1.X**

Drawn by: Irena Gershkovich







Date: 2016-12-22

### Release History:

<b>Rev</b>	<b>Description</b>	<b>Rev by</b>	<b>Check</b>	<b>Concurred</b>	<b>Sys Engr</b>	<b>CM</b>	<b>Date</b>
V1.0	Initial Release	IG					2016-12-22

## Bootloader Overview:

We have recently observed a problem in our test stand that caused the bootloader to unexpectedly erase the main application memory space when the SYNC line was accidentally set high/floating by a faulty ground connection. We will be modifying the bootloader and releasing a FW 3.1 version to address this issue and also make updates to the main application. The following chart describes the communication between the petal controller and the bootloader. The bootloader will now verify that it has received the proper sequence of bootloader commands before erasing the main application memory in preparation for re-programming. The petal controller sends the firmware code via CAN messages to the broadcast positioner id. In order to save time and CAN overhead the positioners keep track of any errors that occur and only respond to the last 'programming verification' command. The process takes about 2 minutes.

Petal Controller Action		Bootloader Action
Set hardware SYNC line high		<b>Power up. If SYNC is high enter programming mode, else jump to main application start address and run main application</b>
Send code size CAN command to broadcast id		<b>Wait for code size CAN command in while loop, incrementing error count if wrong command received, set code_size variable once correct command is received</b>
Send CAN command to specify number of parts that the code has been broken up into (to broadcast id)		<b>Wait for n parts CAN command in while loop, incrementing error count if wrong command received, set n_parts variable once correct command is received</b>
Wait while flash is being erased		<b>Erase Flash</b>
Send all parts and packets with checksums (code is split into parts due to buffer size restriction). Wait between parts while bootloader writes buffer to flash. These commands are sent to broadcast id.		<b>Receive all code parts/packets in for loop, write buffer to flash between parts</b>  Error incremented when: part number/packet number different than expected, packet checksum is different than expected, unexpected CAN command number received
Send request verification command to list of unique ids on given CAN bus (responses by id expected)		<b>If error count is 0, send 'OK' CAN message to petal controller and jump to starting address of main application</b>  Else, send 'ERROR' CAN message to petal controller, erase main application memory and check if SYNC line is still high. If SYNC is held high go through programming process again, if not jump to blank application memory space and do not respond