**FIRMWARE VERSION 2.1:**

* Removed delays in transmit function, otherwise identical to fw version 2.0

**FIRMWARE VERSION 3.0:**

* **Features:**
* Bootloader functionality added
* Firmware now automatically detects when the positioner is finished with a move, move times will no longer need to be added to post-pauses
* ‘SYNC’ has been changed from active-low to active-high
* FW version formatting has been expanded to two bytes
* ADC readout of current monitors has been added (no timing of readout yet)
* Fiducial command has been split into two: one command to select between the fiducial functionality and positioner functionality and a second command to set the duty cycle.
* **Positioner control modifications and considerations:**
  + ‘sudo pip3 install intelhex’ on all petalboxes (this python package is needed for the bootloader)
  + Svn update petalcomm.py and petalcontroller.py
  + Positioner control software now needs to send out the following command via petalcomm (mode can be ‘bootloader’, ‘normal’ or ‘normal\_old’ (‘normal\_old’ mode sends out calculated move times for positioners with FW<3.0):
    - Syntax: pcomm.select\_mode(can\_id, mode = ‘normal\_old’)
    - Positioners with FW<3.0 need to be run in ‘normal\_old’ mode. Positioners with FW3.0 and beyond should be run in ‘normal’ mode (‘normal\_old’ mode is temporary).

*\* If a mixture of positioners with different firmware versions is being tested, ‘normal\_old’ mode will work but the moves will take much longer (as positioners with FW3.0 will pause after move completion in response to the move times still being sent).*

* + To use the bootloader:
    - Power cycle the positioner(s) and then select ‘bootloader’ mode: pcomm.select\_mode(brdcastid, mode = ‘bootloader’)
    - Send the selected firmware hexfile to the positioners (hexfiles will be in petalbox on svn): pcomm.program(brdcastid, hex\_file = ‘fw30.hex’)
    - Verify that programming was successful (send a list of unique CAN ids, not broadcast id): pcomm.request\_verification([3001, 3002, 3003]) Response will be a dictionary with CAN id keys and status values (‘OK’ or ‘ERROR’): {3001 : ‘OK’, 3002: ‘ERROR’, 3003: ‘OK’}
    - After verification, all positioners with ‘OK’ responses will switch into ‘normal’ mode. If an ‘ERROR’ response is received for a given positioner, that positioner will wait for the select\_mode command so that it can be reprogrammed if necessary (by using pcomm.program(unique\_id, mode = ‘bootloader’).