

Feed Update Procedure

This document describes feed update procedure for feed firmware 5.4, that can be done in Hat Creek Radio Observatory. The “feed laptop” has a MPLAB 5.25 software installed (mplab_ide).

Downloading latest code

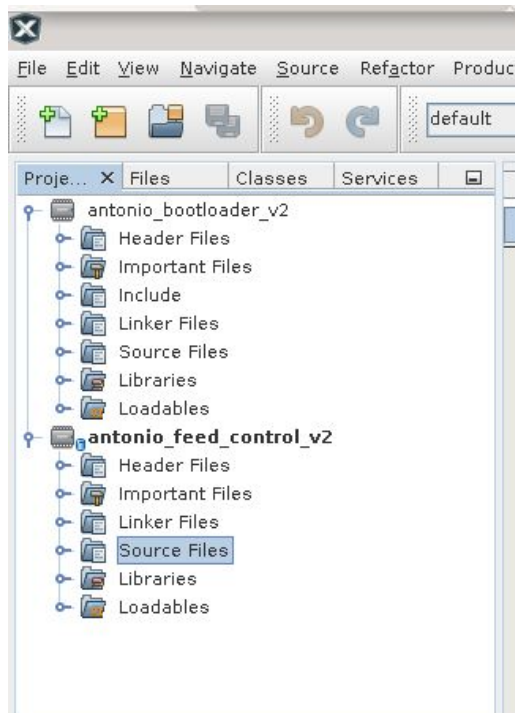
Make sure that the code is in the latest version. On the “feed laptop” the repository resides in

```
/home/sonata/antonio-feed-controller-board
```

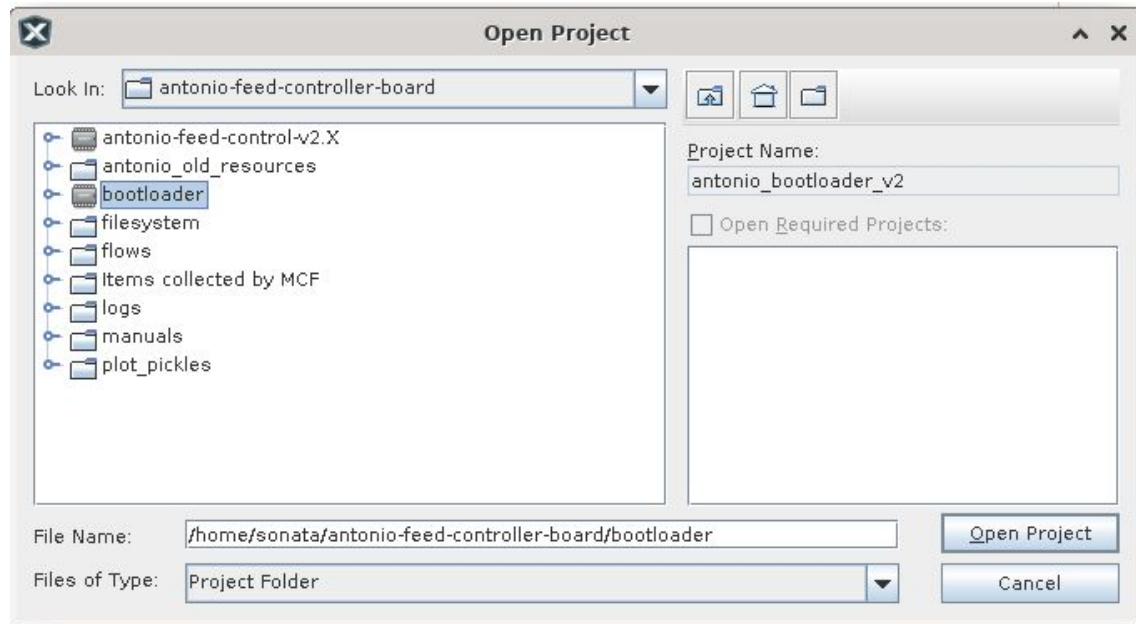
directory. If it's not there or with new system installment, download the repository from the github:

```
git clone https://github.com/SETlatHCRO/antonio-feed-controller-board.git
```

Make sure that both “bootloader” and “feed_control” projects are open. The project tab should look as below (note that **Bold** selection may differ)

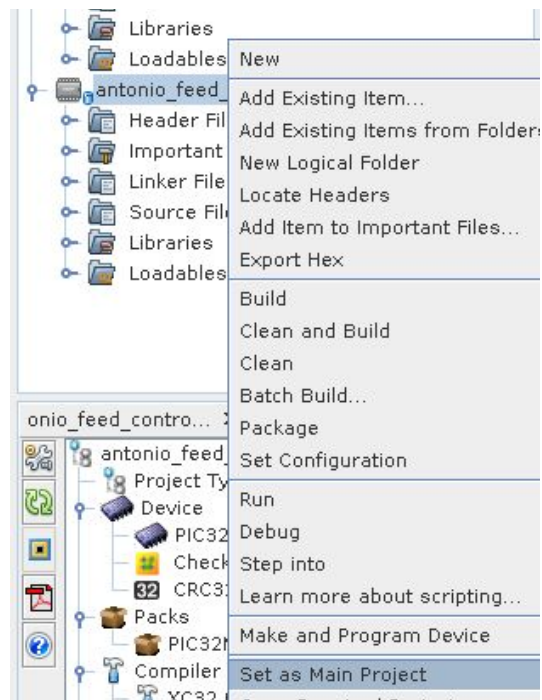


If any of those is missing. Press “File->Open Project ...” and select the missing project. The available projects are displayed with “chipset” icon rather than standard directory icon

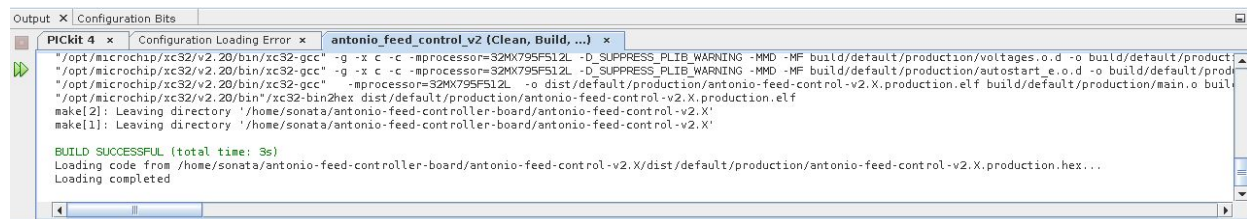


Compile code and prepare image

Since the feed_controller hex file is also copied to the feed “sd” flash, make sure that it’s properly compiled. Right-Click on antonio_feed_control_v2 name and make it the main project



Compile the project. Press “SHIFT + F11” or go to “Production->Clean and Build Project”. The bottom window should look like this:



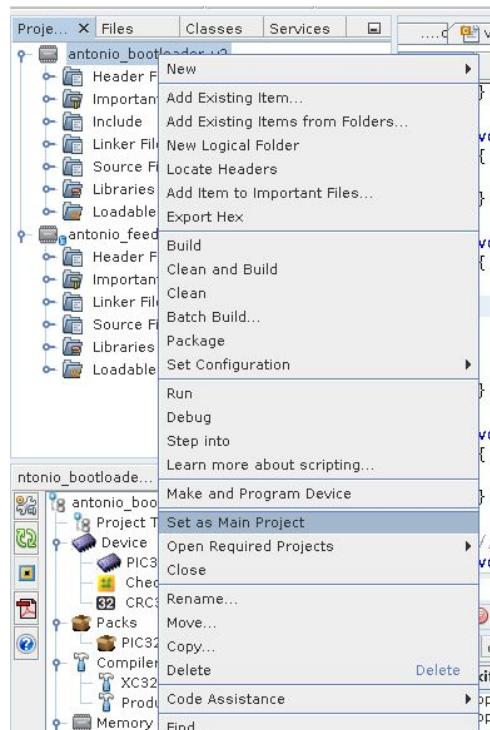
When it's complete, open the terminal in the “filesystem” subdirectory and call bash script

```
./createimage.sh
```

The script may ask for a root password (required to mount loop device). The script should finish with the line “filesystem image created” You may want to move the created image (antoniofsimage) to the main directory.

Compile bootloader

At this point you may want to compile the bootloader project as well. Make sure to set the bootloader as a main program (picture above). Then press “SHIFT+F11” or “Production->Clean and Build Project”. The output of compiler should be as follows (picture below)

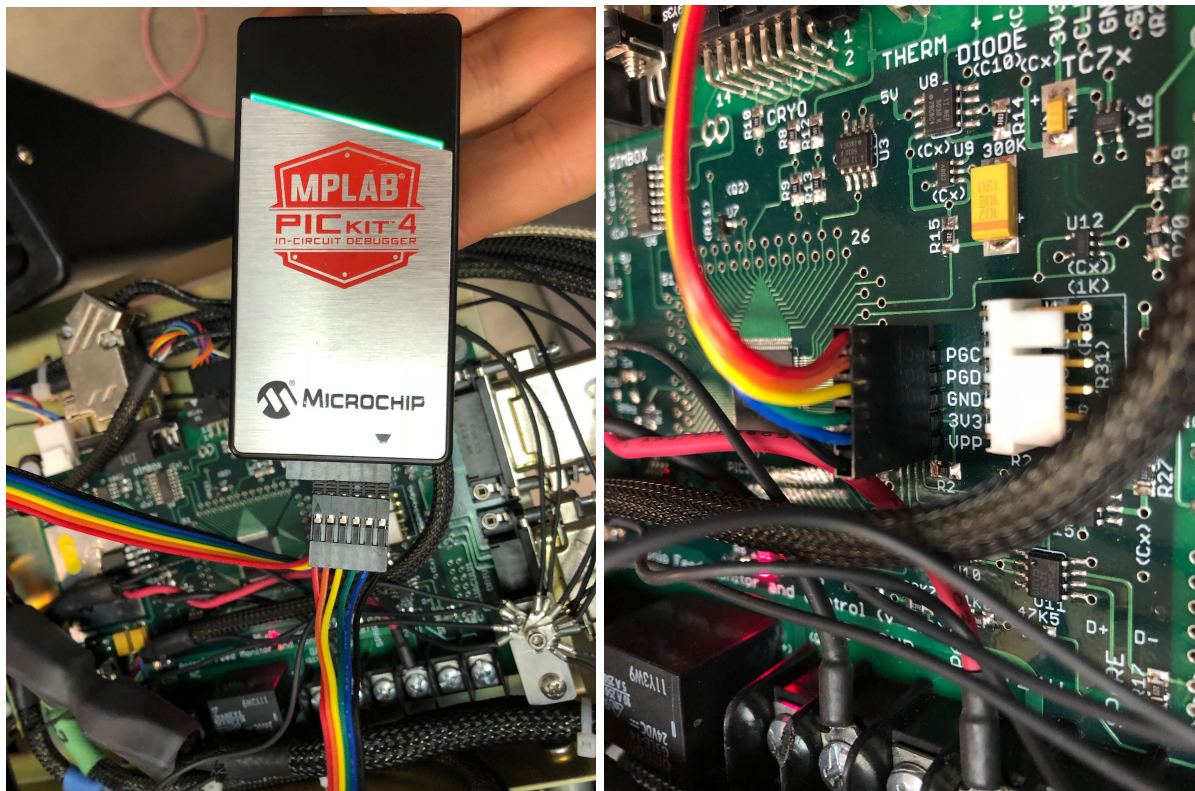


```
PICKIT 4 x Configuration Loading Error x antonio_bootloader_v2 (Clean, Build, ...) x
"/opt/microchip/xc32/v2.20/bin/xc32-gcc" -g -x c -c -mprocessor=32MX795F512L -D SUPPRESS_PLIB_WARNING -I/home/sonata/antonio-feed-controller-board/bootloader/Include -MMD -M
"/opt/microchip/xc32/v2.20/bin/xc32-gcc" -g -x c -c -mprocessor=32MX795F512L -D SUPPRESS_PLIB_WARNING -I/home/sonata/antonio-feed-controller-board/bootloader/Include -MMD -M
"/opt/microchip/xc32/v2.20/bin/xc32-gcc" -mprocessor=32MX795F512L -o dist/default/production/bootloader.production.elf build/default/production/Source/antonio.o build/def
"/opt/microchip/xc32/v2.20/bin/xc32-bin2hex dist/default/production/bootloader.production.elf
make[2]: Leaving directory '/home/sonata/antonio-feed-controller-board/bootloader'
make[1]: Leaving directory '/home/sonata/antonio-feed-controller-board/bootloader'

BUILD SUCCESSFUL (total time: 1s)
Loading code from /home/sonata/antonio-feed-controller-board/bootloader/dist/default/production/bootloader.production.hex...
Loading completed
```

Connecting JTAG

Make sure you have proper JTAG and that it's connected to the USB port. After opening the enclosure, localize the microcontroller board and JTAG pins.

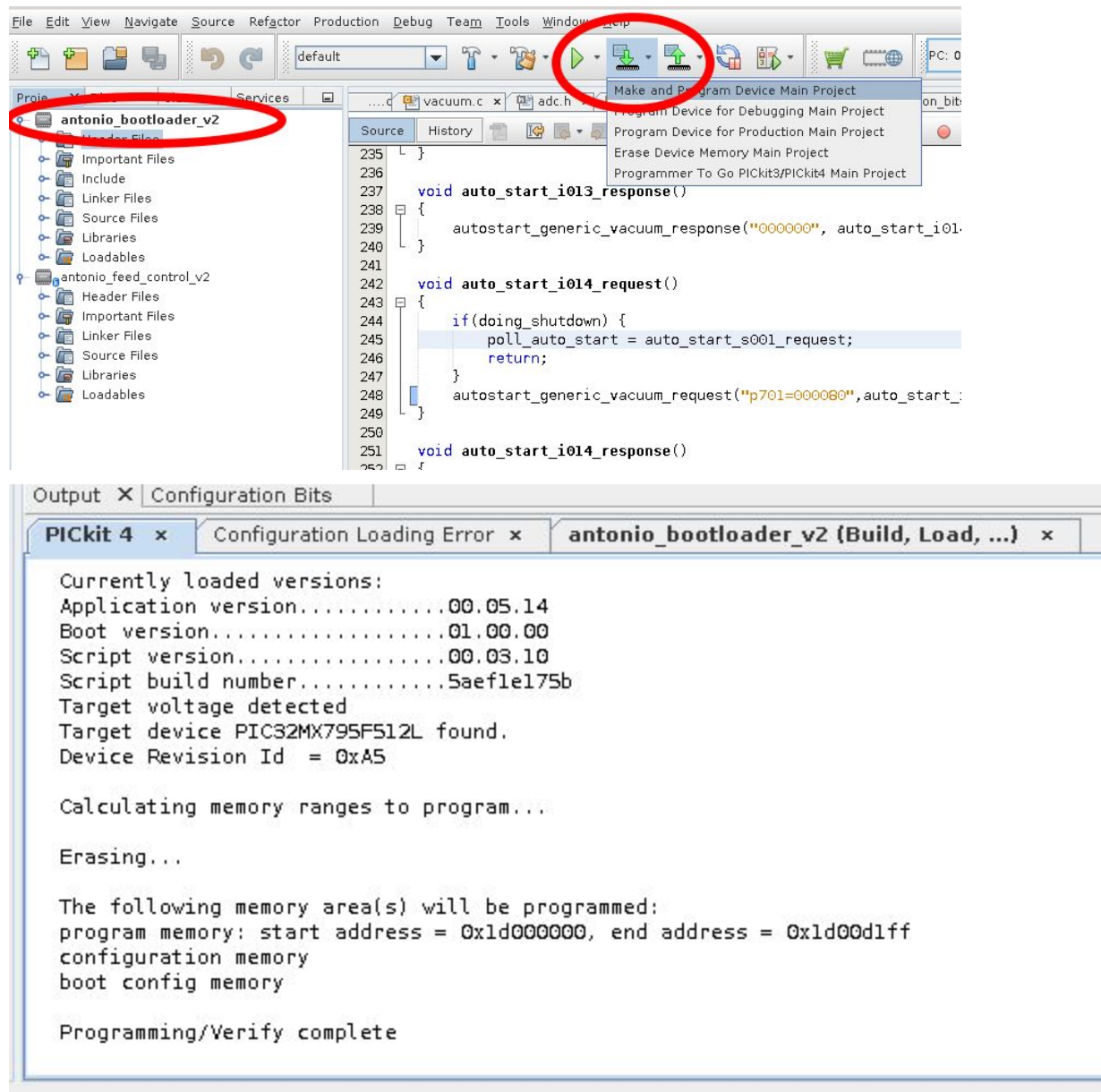


The black cable should be aligned with the vpp pin. Red cable is slack and not connected (note that conector has 6 sockets, where the board has 5 pins).

Make sure that USB-RS232 converter is also connected to your laptop and to the board.

Program controller with the bootloader

Make sure that bootloader is chosen as a “main” program (bold name). Then click on “make and program device main project”. After programming, the successful information should be displayed in the bottom, containing line “Programming/Verify complete”



Upload the image

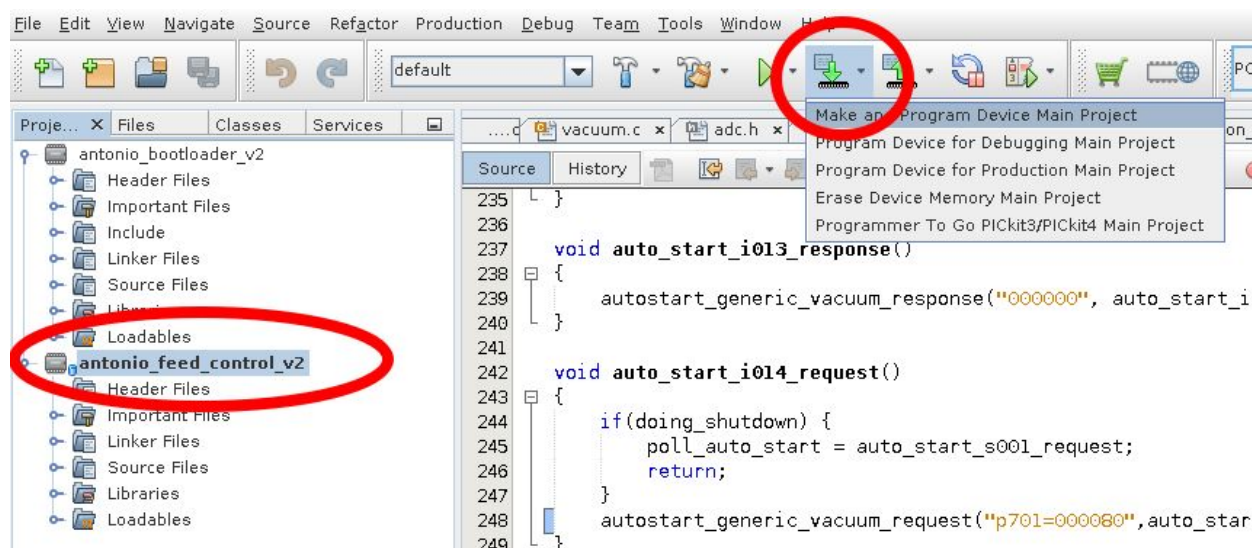
With the bootloader in memory and RS232 connected, make sure that `/dev/ttyUSBX` is visible in the system. You may start uploading the image. Go to main project directory and call “`feed-sd-update.py`” python program, pointing to COM port and fs image, e.g.

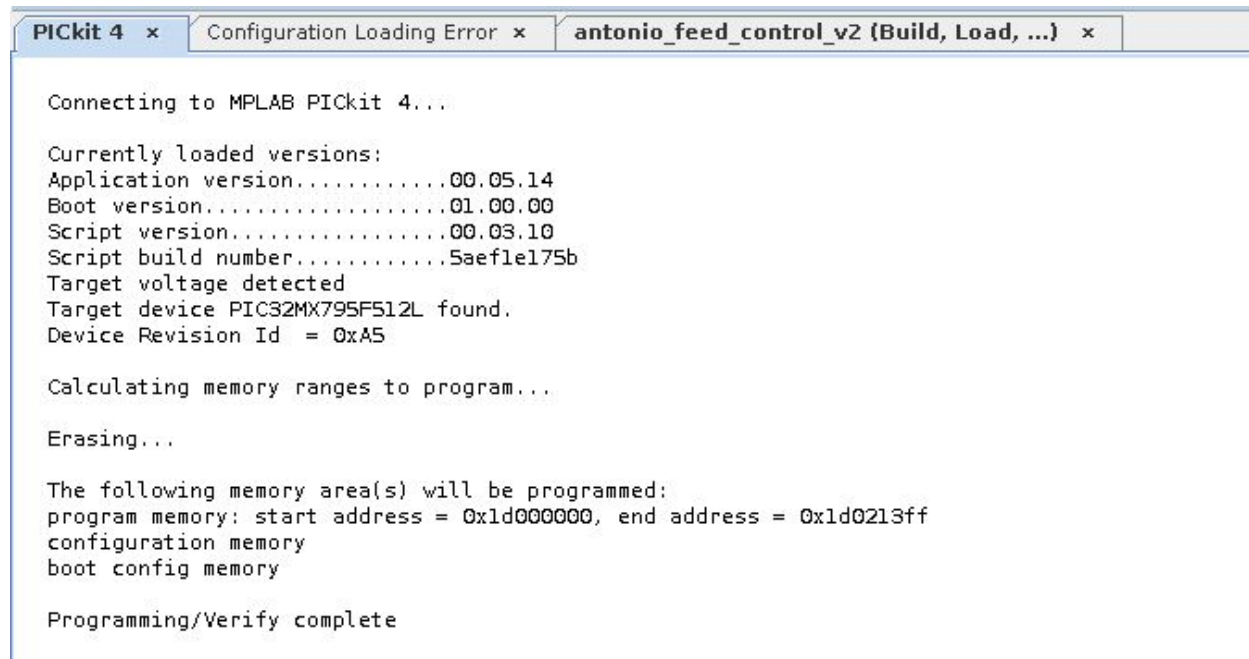
```
./feed-sd-update.py /dev/ttyUSB0 ./filesystem/antoniofsimage
```

The bootloader should reset and slowly accept the image. It may take 10-20 minutes to finish. At the end, you should see the communicat: "feed control board SD update successful - press any key to exit"

Program controller with the control firmware

Make sure that feed control is chosen as a “main” program (bold name). Then click on “make and program device main project”. After programming, the successful information should be displayed in the bottom, containing the line “Programming/Verify complete”. Before programming starts, you may get a pop-up asking if you want to switch PiCkit to that project.





The screenshot shows the MPLAB IDE interface with three tabs: 'PICkit 4', 'Configuration Loading Error', and 'antonio_feed_control_v2 (Build, Load, ...)'. The main window displays the following text:

```
Connecting to MPLAB PICkit 4...

Currently loaded versions:
Application version.....00.05.14
Boot version.....01.00.00
Script version.....00.03.10
Script build number.....5aeffe175b
Target voltage detected
Target device PIC32MX795F512L found.
Device Revision Id = 0xA5

Calculating memory ranges to program...

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x1d000000, end address = 0x1d0213ff
configuration memory
boot config memory

Programming/Verify complete
```

Verify firmware

Open either minicom or the `./feed_test.py` program. Type “getversion” and “man state”. The output should look similar to the picture below. If confirmed, feed is properly programmed. You may want to check “getfeedstartmode”.

getversion
5.4
man state

SYNOPSIS

state
getstate

DESCRIPTION

displays current state of the autostart state machine

OVERVIEW

The states vector is a 32 bit vector. Each state ORs the state vector with its bit. The eight LSBs are responsible for the state itself, whereas higher bits represents error states. Stable low and stable high temperature clears the eight LSB first, before setting the respective bit.

0x000000 - not initialized (or manual autostart)
0x000001 - started initialization
0x000002 - started vacuum pumping
0x000004 - init cooling
0x000008 - cooling down - power
0x000010 - cooling down - temp/stable low temp
0x000020 - heating up
0x000040 - switching off
0x000080 - stable high temp (shutdown) state
0x000100 - e000 occurred (auto start init)
0x000200 - e001 occurred (vac creation)
0x000400 - e002 occurred (vac rot speed error)
0x000800 - e003 occurred (vac power error)
0x001000 - e004 occurred (serious cryo/vac comm error)
0x002000 - e005 occurred (vac rot speed error with cryo running)
0x004000 - e006 occurred (cryo init comm error)
0x008000 - e007 occurred (rot speed on heatup error)
0x010000 - e008 occurred (cold init error)
0x020000 - e009 occurred (vac cold start error)
0x040000 - e010 occurred (vac rpm oscilation)
0x080000 - e011 occurred (cryo motor temp issue)
0x100000 - temp readout problem (A5/A6)
0x200000 - e012 occurred (cryo down while low)
0x400000 - cryo comm problem (timeout)
0x800000 - vac comm problem (timeout)