**Nickel Final Test (FT) Program**

|  |
| --- |
| Table of Contents |
| [Table of Contents 1](#_Toc51248439)  [1. REVISION HISTORY 2](#_Toc51248440)  [2. INTRODUCTION 3](#_Toc51248441)  [3. INSTALLING THE PROGRAM ON A NEW COMPUTER 4](#_Toc51248442)  [4. RUNNING THE PROGRAM 5](#_Toc51248443)  [5. FILES 6](#_Toc51248444)  [6. INDIVIDUAL TESTS 7](#_Toc51248445) |

# REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Date | Author | Contributors | Change Description |
| Rev 1.0 | 09/18/2020 | TRT |  | Original |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# INTRODUCTION

This document describes the Nickel final test (FT) program. The FT program performs the last set of tests before a chip is shipped to customers, whether internal or external. This program therefore has laser coupled illuminated tests that can only be performed after a helmet/mirror assembly has been attached to the chip, and it also has full array tests that can only be performed using a Nickel system at this time. The ‘tester’ is a fully functional Nickel machine, so there are no handlers etc. and all testing is manual.

I lifted code for this program from a number of people, including Dan Frier, Tony Bellofiore, Joe Clark, Zhaoyu He, and Andrew Betts. I have probably missed some people in this list, if I have let me know and the next revision of this document will include your name.

The general requirements for the FT program are:

* A file (trd.csv) that a test engineer can readily change to turn on/off individual tests, set test limits, save/not save images, and stop all testing or not if a chip fails the test.
* A file (cfg.py) that a test engineer can change to set up the program on new testers, add new chip configuration files, and add new product configurations.
* The ability to set the program into a ‘production’ mode where limited information is presented to the operator.
* The ability to perform as many tests on a single device as needed while saving all test data. This allows one to calculated 1st pass yields, retest devices that had bad contact on the first insertion, etc.
* A method for saving data that is clearly marked as engineering or production.
* For each test, save the limits and whether they were applied in the test.
* The output files should have a format that allows additional tests to be added seamlessly.
* The output files should have a format that allows test conditions that vary by product or over time to be saved (test temperatures, integration times, secondary test limits, etc.)
* The output files should be saved in a format that can be examined without special software (csv was chosen here).
* A user interface for examining data and images from previous chip tests, changing the lot/wafer/chip values in case of errors, and changing product/process conditions.
* The ability to rename parameters if the original name was not sufficient for some reason. **On the other hand, the failure\_mode\_bin number assigned to a test can NOT be changed or assigned to another test. The failure\_mode\_bin number is used to sort tests in data analysis code and must (a) remain fixed and (b) be an integer. On the other hand, the order and actual values of the failure\_mode\_bin numbers are arbitrary as long as they are less than 9000.**

# INSTALLING THE PROGRAM ON A NEW COMPUTER

This program runs on Windows 10, and uses python to control the instrument. Do the following steps to install the program on a new computer.

* Copy the latest version of Chewie code on to the Windows 10 computer.
* Connect the Nickel machine to the computer and test that it runs correctly on a chip with Chewie.
* Download Anaconda Installer for Windows.
* Install Python with the Anaconda installer.
* Copy the folder with the tester code, ‘Production\_FT’, on to the desktop of the Windows 10 computer.
* Go to the main folder, Production\_FT, and edit the qsi\_cfg.py file for this machine. You will need to add the name of the new Nickel machine, MOTOR\_ATTENUATOR values for various laser power settings, and the new file paths for this Windows 10 computer.
* Edit the FT.bat file in the ‘Production\_FT’ folder so the paths are correct for the new Windows 10 computer.
* Create a shortcut for the FT.bat file located in the ‘Production\_FT’ directory and move the short cut to the desktop.

# RUNNING THE PROGRAM

Do the following steps to run the program for testing on a Windows 10 machine:

* Insert a chip.
* Double click on the ‘FT’ short cut, which is a short cut to a .bat file.
* Start up the laser if needed. Align the laser if needed.
* Click the ‘Start’ button. The test will run and at the end a pop-up window will say what the bin for this chip is.
* If the chip fails, I recommend re-inserting the chip and running the test again.
* Place the chip in the appropriate tray according to hard bin value.
* Insert more chips as testing progresses.

To change the program into engineering mode, edit qsi\_cfg.py and change HIDE\_CONTROLS = False. On restarting the program more controls will be visible, and in particular the engineer can set how the e-fuse is read, whether the data is saved in an engineering file etc. After all changes have been made, click the ‘Save current state’ button to save the changes. After this set HIDE\_CONTROLS = True to go back into production mode.

# FILES

The program has been divided into multiple files. The hope was that some files could then be re-used as-is for other programs, in particular CP.py. Below is a list of the files and relevant aspects of them. The more important files are listed in **bold type**.

* **FT.py: This is the main program, and is located in /Production\_FT. The GUI interface is created in this file, and code to run the Nickel machine, examine previous test data, and set up for different products or process steps are included here.**
* **qsi\_cfg.py: This file contains settings for changing the test computer, product names, and product configuration files, and is located in the main /Production\_FT directory. Generally, these settings will not get changed very often so I put them into a file that only the test engineer should need to edit.**
* **TRD.csv: These csv files will contain ‘TRD’ in their name and are located in /Production\_FT/trd. For each product type there is an associated ‘TRD’ file, and this association is set up in qsi\_cfg.py. The ‘TRD’ files allow the test engineer to turn on/off individual tests, turn on/off saving of images, set limits for each test, and set whether a test is ‘information\_only’. This is probably the most important file for test engineering. Note that values in the ‘failure\_mode\_bin’ column can NOT be changed for an individual test or transferred over to another test. On the other hand, unused integer values less than 9000 can be added to the column for new tests.**
* **qsi\_ft\_TESTS\_NickelB\_rev0.py: This file has code that loops over all the modules containing individual tests, and is located in /Production\_FT/programs. A python dict, test\_conditions, is defined in this file which has information on running the chip such as temperature set points, parameters for data analysis, and limits for tests that have multiple limit components. The dict test\_conditions is saved for every test.**
* **photonics\_data.csv: This csv file contains information on each lot/wafer such as the presence/absence of an optical filter and the photonics mask files associated with the lot/wafer. Every wafer tested MUST have a line in this file or the test will be stopped.**
* **mask.csv files: The photonics group provides these csv files which indicate what irises, apertures, microlenses, and waveguides are above each pixel. They are located in /Production\_FT/masks. Every lot/wafer must have its associated aperture file in this directory in order to do illuminated tests.**
* qsi\_helpers.py: Various functions that get re-used often in the main program are here, and this file is located in /Production\_FT/utility. This is a ‘rogue’ version of qsi\_helpers.py
* The files below contain code for individual tests and are located in /Production\_FT/modules:
  + qsi\_NickelB\_init\_tests\_rev0.py: This module has tests that are classified as hard bin = 2 (electrical fails).
  + qsi\_NickelB\_vref\_tests\_rev0.py: This module has tests that are classified under hard bin = 3 (cannot adjust ADC vref to spec) and hard bin = 4 (some parts of the array cannot be used).
  + qsi\_NickelB\_dark\_tests\_rev0.py: This module has image quality tests done under dark illumination (hard bin = 4).
  + qsi\_NickelB\_illum\_tests\_rev0.py: This module has image quality tests done under laser illumination (hard bin = 5). Laser alignment and MCLK scans are in this module.
  + nickel\_efuse\_lib.py: A file from Joe that includes the efuse class used for writing/reading the e-fuse. It is located in the ‘utility’ subdirectory of ‘Production\_FT’.
* qsi\_current.csv: This csv file is located in /Production\_FT/utility. It contains information for the current state of the test program like the last lot/wafer/chip tested etc. It provides the ‘memory’ for these variables when the program is stopped.
* current\_config.json: This is the Chewie configuration file being used with the current test, and it gets modified as the test progresses. It is located in /Production\_FT/utility. Other baseline Chewie configuration files are located in /Production\_FT/configurations. Each product has a configuration file associated with it that is set in qsi\_cfg.py.

# INDIVIDUAL TESTS

Individual tests are listed in the table below.

|  | Name | Hard Bin | Soft Bin (FMB) | Units | Notes |
| --- | --- | --- | --- | --- | --- |
| 1 | write\_config | 2 | 5 | none | Write system and chip configuration in json file for this test. |
| 2 | sensor\_ID | 2 | 6 | none | Read the chip type for quick test of SPI read capability. |
| 3-36 | Voltages | 2 | 10-43 | Volts | Chip and system voltages measured with NIM board. |
| 37-57 | Currents | 2 | 60-80 | Amps | Chip and system currents measured with NIM board. |
| 58 | VSUB\_after\_wait | 2 | 81 | Volts | This measurement was added to diagnose VSUB leakage problems and has a 15sec wait time. It will probably be turned off in the near future after this problem is solved. |
| 59 | VSUB\_after\_wait\_I | 2 | 82 | Amps | This measurement was added to diagnose VSUB leakage problems and has a 15sec wait time. It will probably be turned off in the near future after this problem is solved. |
| 60 | write\_efuse | 2 | 90 | none | The e-fuse is written (if desired) and then checked. If the read e-fuse is different from the desired write then this test fails. |
| 61 | read\_efuse | 2 | 91 | none | Whatever is written in the e-fuse is read. This test is information only. |
| 62 | sensor\_temperature\_1 | 2 | 121 | C | Sensor temperature read for information only. |
| 63 | sensor\_temperature\_2 | 2 | 122 | C | Sensor temperature read for information only. |
| 64 | set\_vref\_level | 3 | 311 | none | This test uses code written by Zhaoyu He to adjust the vref\_ctrl register so the reset level is near test\_conditions['vref\_target']. |
| 65 | vref\_ctrl | 3 | 312 | none | The value of vref\_ctrl register that the previous test set. For information only. |
| 66 | vref\_bin0\_5th\_percentile | 3 | 313 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The 5th percentile of the reset image over the entire array is calculated. The region of interest is set by vref\_ROIS in qsi\_cfg.py. Currently a step in row and column of 5 is being used to speed up the calculation. |
| 67 | vref\_bin0\_50th\_percentile | 3 | 314 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The 50th percentile of the reset image over the entire array is calculated. The region of interest is set by vref\_ROIS in qsi\_cfg.py. Currently a step in row and column of 5 is being used to speed up the calculation. |
| 68 | vref\_bin0\_95th\_percentile | 3 | 315 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The 95th percentile of the reset image over the entire array is calculated. The region of interest is set by vref\_ROIS in qsi\_cfg.py. Currently a step in row and column of 5 is being used to speed up the calculation. |
| 69 | vref\_bin0\_stdev | 3 | 316 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The standard deviation of the reset image over the entire array is calculated. The region of interest is set by vref\_ROIS in qsi\_cfg.py. Currently a step in row and column of 5 is being used to speed up the calculation. |
| 70 | vref\_bin1\_5th\_percentile | 3 | 317 | DN | same as vref\_bin0\_5th\_percentile |
| 71 | vref\_bin1\_50th\_percentile | 3 | 318 | DN | same as vref\_bin0\_50th\_percentile |
| 72 | vref\_bin1\_95th\_percentile | 3 | 319 | DN | same as vref\_bin0\_95th\_percentile |
| 73 | vref\_bin1\_stdev | 3 | 320 | DN | same as vref\_bin0\_stdev |
| 74 | vref\_tile\_max\_signal\_bin0 | 3 | 330 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the maximum mean signal is recorded in this test. |
| 75 | vref\_tile\_min\_signal\_bin0 | 3 | 331 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the minimum mean signal is recorded in this test. |
| 76 | vref\_tile\_max\_noise\_bin0 | 3 | 332 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the standard deviation is calculated. The tile with the maximum standard deviation is recorded in this test. |
| 77 | vref\_tile\_min\_noise\_bin0 | 3 | 333 | DN | An image is taken in ‘crop\_raw’ mode to get the reset level. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the standard deviation is calculated. The tile with the minimum standard deviation is recorded in this test. |
| 78 | vref\_tile\_max\_signal\_bin1 | 3 | 334 | DN | same as vref\_tile\_max\_signal\_bin0 |
| 79 | vref\_tile\_min\_signal\_bin1 | 3 | 335 | DN | same as vref\_tile\_min\_signal\_bin0 |
| 80 | vref\_tile\_max\_noise\_bin1 | 3 | 336 | DN | same as vref\_tile\_max\_noise\_bin0 |
| 81 | vref\_tile\_min\_noise\_bin1 | 3 | 337 | DN | same as vref\_tile\_min\_noise\_bin0 |
| 82 | vref\_no\_bad\_tile\_signal\_bin0 | 3 | 340 | none | Tiles with mean signal < test\_conditions['vref\_tile\_min\_signal\_limit'] or mean signal > test\_conditions['vref\_tile\_max\_signal\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 83 | vref\_no\_bad\_tile\_signal\_bin1 | 3 | 341 | none | Tiles with mean signal < test\_conditions['vref\_tile\_min\_signal\_limit'] or mean signal > test\_conditions['vref\_tile\_max\_signal\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 84 | vref\_no\_bad\_tile\_noise\_bin0 | 3 | 342 | none | Tiles with standard deviation < test\_conditions['vref\_tile\_min\_noise\_limit'] or standard deviation > test\_conditions['vref\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 85 | vref\_no\_bad\_tile\_noise\_bin1 | 3 | 343 | none | Tiles with standard deviation < test\_conditions['vref\_tile\_min\_noise\_limit'] or standard deviation > test\_conditions['vref\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 86 | dark\_tint\_min | 4 | 420 | msec | The integration time used for dark minimum integration tests. For information only. |
| 87 | dark\_image\_min\_tint | 4 | 421 | none | Take two frames in dark illumination at dark\_tint\_min integration time. |
| 88 | dark\_min\_tint\_mean\_signal\_bin0 | 4 | 422 | DN | The mean signal over the entire array of one dark illumination frame for bin0. |
| 89 | dark\_min\_tint\_2frame\_mean\_temporal\_noise\_bin0 | 4 | 423 | DN | The Janesick method for calculating mean temporal noise is used. The pixel by pixel difference of 2 dark frames is calculated = diff\_frame.  mean\_temporal\_noise = standard\_deviation(diff\_frame)/sqrt(2) |
| 90 | dark\_min\_tint\_mean\_signal\_bin1 | 4 | 424 | DN | The mean signal over the entire array of one dark illumination frame for bin0. |
| 91 | dark\_min\_tint\_2frame\_mean\_temporal\_noise\_bin1 | 4 | 425 | DN | same as dark\_min\_tint\_2frame\_mean\_temporal\_noise\_bin1 |
| 92 | dark\_min\_tint\_tile\_max\_signal\_bin0 | 4 | 430 | DN | An image is taken under dark illumination with integration time = dark\_tint\_min. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the maximum mean signal is recorded in this test. |
| 93 | dark\_min\_tint\_tile\_min\_signal\_bin0 | 4 | 431 | DN | An image is taken under dark illumination with integration time = dark\_tint\_min. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the minimum mean signal is recorded in this test. |
| 94 | dark\_min\_tint\_tile\_max\_noise\_bin0 | 4 | 432 | DN | Two images are taken under dark illumination with integration time = dark\_tint\_min. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean temporal noise is calculated from the two frame difference using Janesick’s method . The tile with the maximum mean temporal noise is recorded. |
| 95 | dark\_min\_tint\_tile\_min\_noise\_bin0 | 4 | 433 | DN | Two images are taken under dark illumination with integration time = dark\_tint\_min. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean temporal noise is calculated from the two frame difference using Janesick’s method . The tile with the minimum mean temporal noise is recorded. |
| 96 | dark\_min\_tint\_tile\_max\_signal\_bin1 | 4 | 434 | DN | same as dark\_min\_tint\_tile\_max\_signal\_bin0 |
| 97 | dark\_min\_tint\_tile\_min\_signal\_bin1 | 4 | 435 | DN | same as dark\_min\_tint\_tile\_min\_signal\_bin0 |
| 98 | dark\_min\_tint\_tile\_max\_noise\_bin1 | 4 | 436 | DN | same as dark\_min\_tint\_tile\_max\_noise\_bin0 |
| 99 | dark\_min\_tint\_tile\_min\_noise\_bin1 | 4 | 437 | DN | same as dark\_min\_tint\_tile\_min\_noise\_bin0 |
| 100 | dark\_image\_seq\_tint | 4 | 451 | none | Take two frames in dark illumination at test\_conditions['tint\_seq'] integration time. |
| 101 | dark\_seq\_tint\_mean\_signal\_bin0 | 4 | 452 | DN | The mean signal over the entire array of one dark illumination frame for bin0. |
| 102 | dark\_seq\_tint\_2frame\_mean\_temporal\_noise\_bin0 | 4 | 453 | DN | The Janesick method for calculating mean temporal noise is used. The pixel by pixel difference of 2 dark frames is calculated = diff\_frame.  mean\_temporal\_noise = standard\_deviation(diff\_frame)/sqrt(2) |
| 103 | dark\_ seq \_tint\_mean\_signal\_bin1 | 4 | 454 | DN | The mean signal over the entire array of one dark illumination frame for bin0. |
| 104 | dark\_seq\_tint\_2frame\_mean\_temporal\_noise\_bin1 | 4 | 455 | DN | same as dark\_seq\_tint\_2frame\_mean\_temporal\_noise\_bin1 |
| 105 | dark\_ seq \_tint\_tile\_max\_signal\_bin0 | 4 | 460 | DN | An image is taken under dark illumination with integration time = test\_conditions['tint\_seq']. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the maximum mean signal is recorded in this test. |
| 106 | dark\_ seq \_tint\_tile\_min\_signal\_bin0 | 4 | 461 | DN | An image is taken under dark illumination with integration time = test\_conditions['tint\_seq']. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean signal is calculated. The tile with the minimum mean signal is recorded in this test. |
| 107 | dark\_ seq \_tint\_tile\_max\_noise\_bin0 | 4 | 462 | DN | Two images are taken under dark illumination with integration time = test\_conditions['tint\_seq']. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean temporal noise is calculated from the two frame difference using Janesick’s method . The tile with the maximum mean temporal noise is recorded. |
| 108 | dark\_ seq \_tint\_tile\_min\_noise\_bin0 | 4 | 463 | DN | Two images are taken under dark illumination with integration time = test\_conditions['tint\_seq']. The array is divided into 64 ‘tiles’, each served by a given ADC within the chip. For all 64 tiles the mean temporal noise is calculated from the two frame difference using Janesick’s method . The tile with the minimum mean temporal noise is recorded. |
| 109 | dark\_ seq \_tint\_tile\_max\_signal\_bin1 | 4 | 464 | DN | same as dark\_ seq \_tint\_tile\_max\_signal\_bin0 |
| 110 | dark\_ seq \_tint\_tile\_min\_signal\_bin1 | 4 | 465 | DN | same as dark\_ seq \_tint\_tile\_min\_signal\_bin0 |
| 111 | dark\_ seq \_tint\_tile\_max\_noise\_bin1 | 4 | 466 | DN | same as dark\_ seq \_tint\_tile\_max\_noise\_bin0 |
| 112 | dark\_ seq \_tint\_tile\_min\_noise\_bin1 | 4 | 467 | DN | same as dark\_ seq \_tint\_tile\_min\_noise\_bin0 |
| 113 | dark\_min\_tint\_no\_bad\_tile\_noise\_bin0 | 4 | 470 | none | Tiles with mean temporal noise < test\_conditions['dark\_min\_tint\_tile\_min\_noise\_limit'] or mean temporal noise > test\_conditions['dark\_min\_tint\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 114 | dark\_min\_tint\_no\_bad\_tile\_noise\_bin1 | 4 | 471 | none | same as dark\_min\_tint\_no\_bad\_tile\_noise\_bin1 |
| 115 | dark\_min\_tint\_no\_bad\_tile\_signal\_bin0 | 4 | 472 | none | Tiles with mean signal < test\_conditions['dark\_min\_tint\_tile\_min\_noise\_limit'] or mean signal > test\_conditions['dark\_min\_tint\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 116 | dark\_min\_tint\_no\_bad\_tile\_signal\_bin1 | 4 | 473 | none | same as dark\_min\_tint\_no\_bad\_tile\_signal\_bin0 |
| 117 | dark\_seq\_tint\_no\_bad\_tile\_noise\_bin0 | 4 | 474 | none | Tiles with mean temporal noise < test\_conditions['dark\_seq\_tint\_tile\_min\_noise\_limit'] or mean temporal noise > test\_conditions['dark\_seq\_tint\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 118 | dark\_seq\_tint\_no\_bad\_tile\_noise\_bin1 | 4 | 475 | none | same as dark\_seq\_tint\_no\_bad\_tile\_noise\_bin1 |
| 119 | dark\_seq\_tint\_no\_bad\_tile\_signal\_bin0 | 4 | 476 | none | Tiles with mean signal < test\_conditions['dark\_seq\_tint\_tile\_min\_noise\_limit'] or mean signal > test\_conditions['dark\_seq\_tint\_tile\_max\_noise\_limit'] are counted. These are considered to be ‘bad’ tiles. |
| 120 | dark\_seq\_tint\_no\_bad\_tile\_signal\_bin1 | 4 | 477 | none | same as dark\_seq\_tint\_no\_bad\_tile\_signal\_bin0 |
| 121 | dark\_2frame\_dark\_current\_bin0 | 4 | 480 | DN/sec | Two point calculation of dark current from dark\_min\_tint and dark\_seq\_tint frames. f1 = mean signal of min tint frame. f2 = mean signal of seq tint frame.  dark current = (f2 – f1)/(dark\_seq\_tint – dark\_min\_tint) |
| 122 | dark\_2frame\_dark\_current\_bin1 | 4 | 481 | DN/sec | same as dark\_2frame\_dark\_current\_bin1 |
| 123 | dark\_tint1 | 4 | 500 | msec | One of the integration times used for dark integration sweep tests. For information only. |
| 124 | dark\_tint2 | 4 | 501 | msec | One of the integration times used for dark integration sweep tests. For information only. |
| 125 | dark\_tint3 | 4 | 502 | msec | One of the integration times used for dark integration sweep tests. For information only. |
| 126 | dark\_tint4 | 4 | 503 | msec | One of the integration times used for dark integration sweep tests. For information only. |
| 127 | dark\_tint5 | 4 | 504 | msec | One of the integration times used for dark integration sweep tests. For information only. |
| 128 | dark\_scan\_tint | 4 | 510 | none | Sweep the integration time under dark illumination and take test\_conditions['tint\_scan\_frame\_no'] number of frames at each integration time. |
| 129 | dark\_signal\_50th\_bin0\_tint1 | 4 | 520 | DN | The median signal over the entire array for a dark frame taken at dark\_tint1 integration time. |
| 130 | dark\_signal\_50th\_bin0\_tint2 | 4 | 521 | DN | The median signal over the entire array for a dark frame taken at dark\_tint2 integration time. |
| 131 | dark\_signal\_50th\_bin0\_tint3 | 4 | 522 | DN | The median signal over the entire array for a dark frame taken at dark\_tint3 integration time. |
| 132 | dark\_signal\_50th\_bin0\_tint4 | 4 | 523 | DN | The median signal over the entire array for a dark frame taken at dark\_tint4 integration time. |
| 133 | dark\_signal\_50th\_bin0\_tint5 | 4 | 524 | DN | The median signal over the entire array for a dark frame taken at dark\_tint5 integration time. |
| 134 | dark\_signal\_50th\_bin1\_tint1 | 4 | 530 | DN | same as dark\_signal\_50th\_bin0\_tint1 |
| 135 | dark\_signal\_50th\_bin1\_tint2 | 4 | 531 | DN | same as dark\_signal\_50th\_bin0\_tint2 |
| 136 | dark\_signal\_50th\_bin1\_tint3 | 4 | 532 | DN | same as dark\_signal\_50th\_bin0\_tint3 |
| 137 | dark\_signal\_50th\_bin1\_tint4 | 4 | 533 | DN | same as dark\_signal\_50th\_bin0\_tint4 |
| 138 | dark\_signal\_50th\_bin1\_tint5 | 4 | 534 | DN | same as dark\_signal\_50th\_bin0\_tint5 |
| 139 | dark\_temporal\_noise\_50th\_bin0\_tint1 | 4 | 540 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint1 is found. The median standard deviation over all pixels in the array is reported. |
| 140 | dark\_temporal\_noise\_50th\_bin0\_tint2 | 4 | 541 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint2 is found. The median standard deviation over all pixels in the array is reported. |
| 141 | dark\_temporal\_noise\_50th\_bin0\_tint3 | 4 | 542 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint3 is found. The median standard deviation over all pixels in the array is reported. |
| 142 | dark\_temporal\_noise\_50th\_bin0\_tint4 | 4 | 543 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint4 is found. The median standard deviation over all pixels in the array is reported. |
| 143 | dark\_temporal\_noise\_50th\_bin0\_tint5 | 4 | 544 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint5 is found. The median standard deviation over all pixels in the array is reported. |
| 144 | dark\_temporal\_noise\_50th\_bin1\_tint1 | 4 | 550 | DN | same as dark\_temporal\_noise\_50th\_bin0\_tint1 |
| 145 | dark\_temporal\_noise\_50th\_bin1\_tint2 | 4 | 551 | DN | same as dark\_temporal\_noise\_50th\_bin0\_tint2 |
| 146 | dark\_temporal\_noise\_50th\_bin1\_tint3 | 4 | 552 | DN | same as dark\_temporal\_noise\_50th\_bin0\_tint3 |
| 147 | dark\_temporal\_noise\_50th\_bin1\_tint4 | 4 | 553 | DN | same as dark\_temporal\_noise\_50th\_bin0\_tint4 |
| 148 | dark\_temporal\_noise\_50th\_bin1\_tint5 | 4 | 554 | DN | same as dark\_temporal\_noise\_50th\_bin0\_tint5 |
| 149 | dark\_temporal\_noise\_95th\_bin0\_tint1 | 4 | 560 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint1 is found. The 95th percentile over all pixels in the array is reported. |
| 150 | dark\_temporal\_noise\_95th\_bin0\_tint2 | 4 | 561 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint2 is found. The 95th percentile over all pixels in the array is reported. |
| 151 | dark\_temporal\_noise\_95th\_bin0\_tint3 | 4 | 562 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint3 is found. The 95th percentile over all pixels in the array is reported. |
| 152 | dark\_temporal\_noise\_95th\_bin0\_tint4 | 4 | 563 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint4 is found. The 95th percentile over all pixels in the array is reported. |
| 153 | dark\_temporal\_noise\_95th\_bin0\_tint5 | 4 | 564 | DN | The standard deviation of each pixel signal over test\_conditions['tint\_scan\_frame\_no'] number of frames taken at dark\_tint5 is found. The 95th percentile over all pixels in the array is reported. |
| 154 | dark\_temporal\_noise\_95th\_bin1\_tint1 | 4 | 570 | DN | same as dark\_temporal\_noise\_95th\_bin0\_tint1 |
| 155 | dark\_temporal\_noise\_95th\_bin1\_tint2 | 4 | 571 | DN | same as dark\_temporal\_noise\_95th\_bin0\_tint2 |
| 156 | dark\_temporal\_noise\_95th\_bin1\_tint3 | 4 | 572 | DN | same as dark\_temporal\_noise\_95th\_bin0\_tint3 |
| 157 | dark\_temporal\_noise\_95th\_bin1\_tint4 | 4 | 573 | DN | same as dark\_temporal\_noise\_95th\_bin0\_tint4 |
| 158 | dark\_temporal\_noise\_95th\_bin1\_tint5 | 4 | 574 | DN | same as dark\_temporal\_noise\_95th\_bin0\_tint5 |
| 159 | dark\_current\_bin0 | 4 | 580 | DN/sec | Fit dark\_signal\_50th\_bin0\_tint1 etc. as a function of dark\_tint1 etc. to a line. The slope of this line is the dark current. |
| 160 | dark\_current\_bin1 | 4 | 581 | DN/sec | Fit dark\_signal\_50th\_bin1\_tint1 etc. as a function of dark\_tint1 etc. to a line. The slope of this line is the dark current. |
| 161 | diff\_image\_dark\_current\_bin0 | 4 | 582 | none | Subtract the dark\_tint1 image from dark\_tint5 image and save. This difference image can be useful for debugging dark current hot spots in the array. |
| 162 | diff\_image\_dark\_current\_bin1 | 4 | 583 | none | same as diff\_image\_dark\_current\_bin0 |
| 163 | read\_noise\_bin0 | 4 | 590 | DN | Fit (dark\_temporal\_noise\_50th\_bin0\_tint1)^2 etc. as a function of dark\_tint1 etc. to a line. The intercept of this line is read\_noise\_bin0. |
| 164 | read\_noise\_bin1 | 4 | 591 | DN | Fit (dark\_temporal\_noise\_50th\_bin1\_tint1)^2 etc. as a function of dark\_tint1 etc. to a line. The intercept of this line is read\_noise\_bin1. |
| 165 | dark\_image\_noise\_min\_tint\_bin0 | 4 | 592 | none | The standard deviation of the signals for each pixel in the frames taken at dark\_tin1 are calculated and put into a new array. The image of this temporal noise image is saved. |
| 166 | dark\_image\_noise\_min\_tint\_bin1 | 4 | 593 | none | same as dark\_image\_noise\_min\_tint\_bin0 |
| 167 | dark\_image\_noise\_max\_tint\_bin0 | 4 | 594 | none | The standard deviation of the signals for each pixel in the frames taken at dark\_tin5 are calculated and put into a new array. The image of this temporal noise image is saved. |
| 168 | dark\_image\_noise\_max\_tint\_bin1 | 4 | 595 | none | same as dark\_image\_noise\_max\_tint\_bin0 |
| 169 | CGC\_bin0 | 4 | 596 | electrons/DN | Fit (dark\_temporal\_noise\_50th\_bin0\_tint1)^2 etc. as a function of dark\_tint1 etc. to a line. S = slope of this line fit in DN^2/msec. CGC\_bin0 = dark\_current/slope |
| 170 | CGC\_bin1 | 4 | 597 | electrons/DN | same as CGC\_bin0 |
| 171 | dark\_current\_electrons\_bin0 | 4 | 598 | electrons/sec | dark\_current\_electrons\_bin0 = (dark\_current\_bin0)(CGC\_bin0) |
| 172 | dark\_current\_electrons\_bin1 | 4 | 598 | electrons/sec | dark\_current\_electrons\_bin0 = (dark\_current\_bin1)(CGC\_bin1) |
| 173 | read\_noise\_electrons\_bin0 | 4 | 600 | electrons | read\_noise\_electrons\_bin0 = (read\_noise\_bin0)(CGC\_bin0) |
| 174 | read\_noise\_electrons\_bin1 | 4 | 601 | electrons | read\_noise\_electrons\_bin1 = (read\_noise\_bin1)(CGC\_bin1) |
| 175 | align\_laser | 10 | 611 | none | Scan MOTOR\_X, MOTOR\_Y, MOTOR\_THETA\_Y, tint, MOTOR\_ATTEN, and MCLK to find the maximum laser coupling. |
| 176 | beam\_steer\_X | 5 | 612 | none | The MOTOR\_X position found after laser alignment. |
| 177 | beam\_steer\_Y | 5 | 613 | none | The MOTOR\_Y position found after laser alignment. |
| 178 | beam\_steer\_THETA\_X | 5 | 614 | none | The MOTOR\_THETA\_X position found after laser alignment. For information only. |
| 179 | beam\_steer\_THETA\_Y | 5 | 615 | none | The MOTOR\_THETA\_Y position found after laser alignment. |
| 180 | beam\_steer\_ATTEN | 5 | 616 | none | The MOTOR\_ATTENUATOR position found after laser alignment. |
| 181 | beam\_steer\_ICW | 5 | 617 | none | The MOTOR\_ICW position found after laser alignment. For information only. |
| 182 | beam\_steer\_ROLL | 5 | 618 | none | The MOTOR\_ROLL position found after laser alignment. For information only. |
| 183 | set\_laser\_power\_seq\_tint | 5 | 620 | none | With the integration time started at test\_conditions['tint\_seq'], scan the laser power to bring the median illuminated pixel signal to test\_conditions['illum\_median\_signal']. If the signal can not be set to this target lower the integration time and try again. This test is not performed on devices with no optical filters as there is still intense laser scatter at the minimum laser power and tint settings generally. |
| 184 | illum\_median\_signal | 5 | 621 | none | The illuminated median signal over pixels found in test 620. |
| 185 | illum\_median\_laser\_power | 5 | 622 | mW | The laser power found in test 620. |
| 186 | illum\_median\_tint | 5 | 623 | msec | The integration time found in test 620. |
| 187 | illum\_median\_laser\_power\_x\_tint | 5 | 624 | mW\_msec | Laser power x integration time x (median signal)/(target signal) found in test 620. |
| 188 | illum\_image\_seq\_tint | 5 | 625 | none | Take an image under illumination and save. |
| 189 | MCLK\_scan | 5 | 630 | none | Set MCLK=0 and adjust the machine so the median pixel signal is test\_conditions['illum\_median\_signal']. For chips with optical filters this should already be done. For chips without optical filters the laser scatter will be reduced by adjusting MOTOR\_Y. After the signal level has been adjusted, scan MCLK and save the median signal over pixels with apertures in an array. |
| 190 | MCLK\_knee\_bin0 | 5 | 640 | MCLK\_units | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. The MCLK\_knee value is reported. |
| 191 | MCLK\_knee\_bin1 | 5 | 641 | MCLK\_units | same as MCLK\_knee\_bin0 |
| 192 | MCLK\_rej\_1nsec\_bin0 | 5 | 642 | none | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. Using the fitted function calculate the ratio of the function value at MCLK = 0 divided by MCLK 1 nsec offset from MCLK\_knee. |
| 193 | MCLK\_rej\_1nsec\_bin1 | 5 | 643 | none | same as MCLK\_rej\_1nsec\_bin0 |
| 194 | MCLK\_rej\_0p5nsec\_bin0 | 5 | 644 | none | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. Using the fitted function calculate the ratio of the function value at MCLK = 0 divided by MCLK 0.5 nsec offset from MCLK\_knee. |
| 195 | MCLK\_rej\_0p5nsec\_bin1 | 5 | 645 | none | same as MCLK\_rej\_0p5nsec\_bin0 |
| 196 | MCLK\_rej\_0p25nsec\_bin0 | 5 | 646 | none | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. Using the fitted function calculate the ratio of the function value at MCLK = 0 divided by MCLK 0.25 nsec offset from MCLK\_knee. |
| 197 | MCLK\_rej\_0p25nsec\_bin1 | 5 | 647 | none | same as MCLK\_rej\_0p25nsec\_bin0 |
| 198 | MCLK\_amp\_bin0 | 5 | 648 | DN | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. The MCLK\_amp value is reported. |
| 199 | MCLK\_amp\_bin1 | 5 | 649 | DN | same as MCLK\_amp\_bin0 |
| 200 | MCLK\_bgnd\_bin0 | 5 | 650 | DN | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. The MCLK\_bgnd value is reported. |
| 201 | MCLK\_bgnd\_bin1 | 5 | 651 | DN | same as MCLK\_bgnd\_bin0 |
| 202 | MCLK\_tau\_bin0 | 5 | 652 | none | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. The MCLK\_tau value is reported. |
| 203 | MCLK\_tau\_bin1 | 5 | 653 | none | same as MCLK\_tau\_bin0 |
| 204 | MCLK\_res\_bin0 | 5 | 654 | none | Fit bin0 median pixel signal as a function of MCLK setting to the function given in the footnote below. The MCLK\_res value is reported. |
| 205 | MCLK\_res\_bin1 | 5 | 655 | none | same as MCLK\_res\_bin0 |
| 206 | MCLK\_diff | 5 | 656 | none | MCLK\_diff = MCLK\_knee\_bin1 - MCLK\_knee\_bin0 |

Footnote:

The function MCLK scans are fit to is:

convolved with a Gaussian resolution function:

The convolved fit function is:

where