**RMD – HSFL Version FSW Test Commands**

This document will be a brief description of the commands to issue for collecting data with the Hawaii Instrument.

[1. Format Requirements 2](#_Toc27581351)

[2. Testing Commands 3](#_Toc27581352)

[2.1 System Parameter Loading 3](#_Toc27581353)

[2.2 Begin a DAQ Science Run 4](#_Toc27581354)

[2.3 ID and Run Numbers during a Science Run 6](#_Toc27581355)

[2.4 End the DAQ Science Run 6](#_Toc27581356)

[2.5 Data Transfer Mode and Parameters 7](#_Toc27581357)

[2.6 Waveform Collection 8](#_Toc27581358)

[2.7 Notes for Further DAQ Science Runs 9](#_Toc27581359)

[3. HSFL Test Command Set 10](#_Toc27581360)

[4. ASU Testing Command Set 12](#_Toc27581361)

[5. Appendix 15](#_Toc27581362)

[5.1 APID Values for CCSDS Packets 15](#_Toc27581363)

[5.2 Commanded Functions That Return a Packet Other Than SUCCESS or FAILURE 15](#_Toc27581364)

# Format Requirements

Commands are sent to the instrument as ASCII statements. Bytes received by the RS422 are interpreted as ASCII characters.

All commands have parameters which follow the command. Those parameters can be either integers, floats, or strings, as documented below. All parameters are required, and will be parsed as the expected data type. Any input given to the command will be parsed as the data type given. If the number of parameters incorrect, the command will be rejected with a Command Failure packet (0x01).

**The parsing relies on the “\_” (underscore) character.**

Parameters and naming sequences not separated by the underscore character will not be recognized by the system.

The number which follows almost all the commands is the Detector Number (0 or 1) indicated as “#” (pound) in the command list below. The Mini-NS is comprised of two separate boards and commands need to be addressed to one or the other board.

NOTE: The HSFL Neutron-1 only has one Detector Number because the system is comprised of just one board. The Detector Number for the HSFL board is

**Detector Number = 0**

Each board will parse, but **silently reject** commands which are addressed to the other board.

**Commands must be terminated with a newline “\n” character.**

On the completion of a command, the return value will be written to the RS422 wrapped in a CCSDS packet; the possible return values are listed below each command. The success or failure of a command is indicated via the APID bytes (ie. the type of packet returned), see Table 2, in the appendix. All commanded functions except MNS\_START return a CCSDS packet upon completion, but some commanded functions, see Table 3, return a packet other than SUCCESS/FAILURE (eg. GETSTAT returns an SOH packet instead of a SUCCESS packet).

# Testing Commands

These commands assume that the board is powered and in Standby Mode.

## System Parameter Loading

Parameters to set and command strings to set those parameters:

Set the Trigger Threshold

MNS\_TRG\_0\_8850\n

The trigger threshold of 8850 is what we used with the testing and validation software.

Integration times

MNS\_INT\_0\_-52\_88\_472\_6000\n

These are the typical integration times and are times in nanoseconds.

HV pot values

MNS\_ENABLE\_ACT\_0\n

This enables the active components of the Mini-NS.

MNS\_HV\_0\_1\_196\n

This sets the HV on the high voltage potentiometer attached to channel 1 to -766.8 Volts.

Energy calibration

MNS\_ECAL\_0\_1.0\_0.0\n

This gives us no energy calibration, which is acceptable for this test

Neutron cuts:

**NOTE:** *As of 10/22/2019 setting these values is not necessary and may result in strange count values for the Counts Per Second data product.*

To set the regular box cuts:

MNS\_NGATES\_0\_0\_0\_288450\_384600\_0.2\_0.4\n

To set the wide box cuts:

MNS\_NGATES\_0\_0\_1\_269220\_403830\_0.16\_0.44\n

There are two “boxes” which we can move around using these cuts to define an area which is “neutron sensitive”.

NOTE: The values for the neutron cut windows are wide and may catch more than just neutrons. This is intentional and will only affect the number of counts we see in the CPS files and what is reported by the SOH. We would expect these values to be above what would normally be expected (due to a larger window allowing in more counts).

Once each value is set, the system has an internal configuration file which will be updated. After receiving a command success from one of these commands, the values is now set within the system until is updated again. Thus, the user need only call these functions once each time that it needs to be set.

## Begin a DAQ Science Run

Move the system to DAQ mode:

MNS\_DAQ\_0\_1\n

This gives the DAQ run an ID number of “1” which the user must keep track of.

**NOTE**: an important note is that when MNS\_DAQ is called, it loops trying to find a unique combination of ID and run numbers so that it can create a new directory to store the data products in. The system will return a Command Success packet containing the folder name for that run which follows the format shown in section 2.4 below. This is how we know what the run number is. The other way is to keep track of how many times MNS\_DAQ\_ is called. The run number increments each time MNS\_DAQ is called.

Format of the Command Success Packet returned from the DAQ command:

|  |  |  |  |
| --- | --- | --- | --- |
| Byte | Description | Data Type | Group |
| 0-3 | Sync Marker | Byte | Primary CCSDS Header |
| 4 | Flags and Detector Number |
| 5 | APID |
| 6 | Group Flags and Sequence Count MSB |
| 7 | Sequence Count LSB |
| 8 | Packet Length MSB |
| 9 | Packet Length LSB |
| 10 | Reset Request Flag | Secondary CCSDS Header |
| 11-… | ASCII String: Commanded Input | ASCII String | Data |
| … | “\n” | Signed Char |
| … | ASCII String: Folder Name | ASCII String |
|  | Simple Checksum | Byte | RMD Data Checksums |
|  | Fletcher Checksum |
|  | CCSDS Checksum MSB | Checksum |
|  | CCSDS Checksum LSB |

Currently, what is returned is the command which was sent by the flight computer (the commanded input), a newline character, and the folder name for the data acquisition run. Because this was missed when changing over to fixed field format, the fields labeled as Commanded Input will grow with the input command. The ASCII string labeled Folder Name will always be the same size, and the format of that string is in section 2.4 below.

On the next page, Table 1 is an example of what a command success packet returned from a MNS\_DAQ command would look like. This assumes the following commanded input:

MNS\_DAQ\_0\_1

Table 1: Example Command Success Packet

|  |  |  |
| --- | --- | --- |
| Byte | Description | Group |
| 0 | 0x35 | Sync Marker |
| 1 | 0x2E |
| 2 | 0xF8 |
| 3 | 0x53 |
| 4 | 0x0A | Primary CCSDS Header |
| 5 | 0x00 |
| 6 | 0xC0 |
| 7 | 0x01 |
| 8 | 0x00 |
| 9 | 0x1E |
| 10 | 0x00 | Secondary CCSDS Header |
| 11 | 0x4D | Mini-NS Data Bytes |
| 12 | 0x4E |
| 13 | 0x53 |
| 14 | 0x5F |
| 15 | 0x44 |
| 16 | 0x41 |
| 17 | 0x51 |
| 18 | 0x5F |
| 19 | 0x30 |
| 20 | 0x5F |
| 21 | 0x31 |
| 22 | 0x0A |
| 23 | 0x30 |
| 24 | 0x3A |
| 25 | 0x2F |
| 26 | 0x49 |
| 27 | 0x30 |
| 28 | 0x30 |
| 29 | 0x30 |
| 30 | 0x31 |
| 31 | 0x5F |
| 32 | 0x52 |
| 33 | 0x30 |
| 34 | 0x30 |
| 35 | 0x30 |
| 36 | 0x31 |
| 37 | Cs1 | RMD Simple Checksum |
| 38 | Cs2 | RMD Fletcher Checksum |
| 39 | Cs3 | CCSDS Checksum MSB |
| 40 | Cs4 | CCSDS Checksum LSB |

NOTE: the checksums will changed based on the contents of the packet and may be different even if the packet is very similar to one previously sent. Thus, the values should be calculated each time a packet is received.

Begin the run with:

MNS\_START\_0\_realTime\_timeOut\n

RealTime should be a time from the flight computer. For now, we can use any unsigned long long value.

timeOut gives a value, in units of minutes, after which the system will break out of a DAQ run. For a value of timeOut=2, when the system has looped for 120 seconds the run will be gracefully ended with no input.

## ID and Run Numbers during a Science Run

The SOH packets have been updated to include the ID and Run numbers for the DAQ run. The ID Number and Run Number fields are only relevant when the SOH reports that it is in Pre-DAQ, DAQ, or WAV mode. The ID and Run numbers tell the user what the file and folder names for the DAQ or WAV runs are.

When in Pre-DAQ or DAQ mode, both the ID and Run numbers will be reported.

When in WAV mode, the ID number will be reported, but the Run number will always be set to 0.

When in any other mode, the ID and Run numbers will be set to 0.

## End the DAQ Science Run

There are three options for ending a DAQ run:

* Timeout

No input necessary

Just wait the number of minutes specified to the detector via the START command.

* Ending the DAQ run

MNS\_END\_0\_realTime\n

RealTime should be a time from the flight computer. For now, we can use any unsigned long long value.

* Breaking out of the process:

MNS\_BREAK\_0

Leave the DAQ process, but specify no real time.

Choose one of the above three options (time out, END, BREAK) and send the command (or wait), the system will finalize the run and return to standby mode.

## Data Transfer Mode and Parameters

Using the values from the previous DAQ run, we first want to identify the folders/files that were created so we can properly request the files back. The folders are created via the following formula:

“0:/I0000\_R0000”

* The “0:/” specifies the root directory on the SD card.
* “I” is an uppercase letter i, which stands for ID Number. This is the unique identifier that the user sends with the MNS\_DAQ\_ command
* “0000” is the ASCII representation of the ID Number.
* “R” is an uppercase letter r, which stands for Run Number. This is an internal number that is tracked by the detector and incremented each time a DAQ run is started during a power cycle.
* “0000” is the ASCII representation of the Run Number

As an example of a folder name: if we choose and specify an ID Number of 12 and the this is the third time we have started a DAQ run during this particular power cycle we would have a folder name of:

“0:/I0012\_R0003”

With the knowledge of a file the user wants to transfer and its name, the user can construct the parameters to give to transfer so the proper file is returned.

Choose a file type to request:

* 5 = counts per second file
* 6 = Waveforms
* 7 = event-by-event file
* 8 = 2D histogram from PMT 0
* 11 = 2D histogram from PMT 1
* 12 = 2D histogram from PMT 2
* 13 = 2D histogram from PMT 3

This file type goes into the file type parameter

Then place the ID and Run numbers into their respective fields

The values for the set number are only important for the event-by-event data product, for all other data products, these values are 0.

To specify the set number low, start at 0 and transfer files until a command failure is returned, this indicates there are no more files from that sequence. The number of files generated by event-by-event data can vary, but can be up to and more than 75 for a long run.

At the moment, set number high is not used, so always set this to 0

For example, to get the counts per second data:

MNS\_TX\_0\_5\_ID\_Run\_0\_0

Place the integer values for the ID and Run number in. The ID should be known, the run number is the number of times the DAQ command has been issued during the current power cycle. The set numbers should be 0.

To get the 2D histogram from PMT 1:

MNS\_TX\_0\_8\_ID\_Run\_0\_0

To get the first EVT data product set file:

MNS\_TX\_0\_7\_ID\_Run\_0\_0

And then…

MNS\_TX\_0\_7\_ID\_Run\_1\_0 and so forth.

## Waveform Collection

Prepare the system to take diagnostic waveform data and initialize the WAV module.

Move the system to WAV mode:

MNS\_WF\_0\_0\_5\_10\n

This gives the WAV collection an ID number of “10” and requests for 5 waveform buffers of adjacent average waveforms to be collected.

There is currently one type of waveform collection supported, adjacent average (WFType = 0), so parameter 2, WFType should always be set to 0.

The system is placed in a new polling loop and SOH indicates DAQ (Waveform collection) Mode.

The system will stop collecting waveforms when the user requested number of waveform buffers (parameter 3) has been acquired or when the BREAK command is received.

## Transfer the SD Card Directory Contents

As the system performs data collections, either via DAQ or WF runs, folders and data product files will be created and stored on the SD cards. The folders are named following the syntax described in Section 2.5. The files are named with the following sytax:

* EVT
  + “evt\_S0000.bin”
  + This file is named differently for the current iteration of the FSW to allow for it to be broken up into smaller chunks
  + Since this data product is often very large ( > 1 MB), by chunking it up we may transfer the file more conveniently, at the expense of simplicity in naming, organization
  + Future iterations may see this file being compacted to a single file
  + The option to change the chunk size for this data product must be “baked in” to the FSW, it is currently set to 1 MB
* CPS
  + “cps.bin”
* 2DH
  + “2dX.bin”
  + There are four variations of this file name string, one for each of the PMTs. Those strings will appear as the following:
    - “2d0.bin”
    - “2d1.bin”
    - “2d2.bin”
    - “2d3.bin”

When the MNS\_DIR command is sent, it makes a request the system to scan the SD cards and report all the files and folders currently present on either SD card 0 (primary) or SD card 1 (backup). The system will transfer a number of packet whose contents are a list of all files stored on the instrument’s SD card. This will not report hidden files or folders or system files or folders, such as SD card system files.

The packets will list each of the following items for each folder on the Mini-NS:

* Folder name
* File 1 name
* File 1 size
* File 2 name
* File 2 size
* …

Each of the files listed will have a 4 byte unsigned integer field which follows it which tells what the file size is for that file at the time of the scan.

Files are listed after their containing folder. There are 3 files which are an exception to this, they are the following files:

* MNSLOG.txt
* MNSCONF.bin
* SD0.txt or SD1.txt

Each of these files are system files and are stored on the root directory “0:/” and are not listed with the root directory. They should be assumed to be on the root directory.

## Notes for Further DAQ Science Runs

**The Mini-NS must be power cycled after making any data acquisition or waveform run!**

There is a “false” event generated the first time that DAQ is called each time the board is power cycled. Thus, each subsequent DAQ run that power cycle will not have a “false” event present in the data products.

The log and configuration files can’t be transferred with this release.

# HSFL Test Command Set

The following table shows a set of commands that can be issued to the system to set the system parameters, take data, and transfer the data out.

|  |  |
| --- | --- |
| Command | Description |
| MNS\_TRG\_0\_8850\n | Set trigger threshold to 8850 |
| MNS\_INT\_0\_-52\_88\_472\_6000\n | Set integration times |
| MNS\_ENABLE\_ACT\_0\n | Enable the 3.3 V |
| MNS\_HV\_0\_3\_196\n | Set the high voltage potentiometer to -766.8 Volts |
| MNS\_ECAL\_0\_1.0\_0.0\n | Set energy calibration parameters |
| MNS\_DAQ\_0\_1 | Change system to data acquisition mode |
| MNS\_START\_0\_123456\_5 | Begin data acquisition; take data for 5 minutes |
| MNS\_END\_0\_654321 | End data acquisition |
| MNS\_TX\_0\_5\_1\_1\_0\_0 | Transfer CPS data product file |
| MNS\_TX\_0\_7\_1\_1\_0\_0 | Transfer EVT data product file |
| MNS\_TX\_0\_8\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 1 |
| MNS\_TX\_0\_11\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 2 |
| MNS\_TX\_0\_12\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 3 |
| MNS\_TX\_0\_13\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 4 |
| Power Cycle |  |
| MNS\_WF\_0\_0\_5\_1 | Begin a Waveform taking run requesting 5 waveforms |
| MNS\_TX\_0\_6\_1\_0\_0\_0 | Transfer WAV data product file |
| MNS\_DIR\_0\_0 | Report the contents of SD Card 0 |
| MNS\_DIR\_0\_1 | Report the contents of SD Card 1 |

NOTE: If the system does not have an analog board attached, the MNS\_HV commanded function will hang the system.

For the commands above, the following variables are identified:

* Detector Number = 0
* DAQ ID Number = 1
* DAQ Run Number = 1
* DAQ START real time = 123456
* DAQ Time out = 5 minutes
* DAQ END real time = 654321
* WAV Type = 0
* WAV Number of Waves = 5
* WAV ID number = 1

For the file transfers the end result should be:

* CPS: 3 packets
* EVT: 521 packets
* 2DH: 8 packets
* WAV:

The files on the SD card should be the following sizes (for a 5 minute run):

* CPS: 4,556 bytes
* EVT: 1,048,608 bytes – for a set file
  + Each EVT data product is split into many set files, each file is 1 MB plus a header
  + The final EVT file will be smaller
* 2DH: 15,808 bytes – for each PMT
* WAV:

The files on the SD card should be the following sizes (for a 30 minute run):

* CPS: 25,444 bytes
* EVT: 1,048,608 bytes – for the first and intermediate set files
  + Each EVT data product is split into many set files, each file is 1 MB plus a header
* EVT = 228 < size < 1,048,608 bytes – for the final set file
* 2DH: 15,808 bytes – for each PMT

When transferring out the data products after a data acquisition run, the number of packets transferred per file transferred will be the following:

* CPS = 13 packets
* EVT = 529 packets
* 2DH = 8 packets

NOTE: if the event rate of the source is low, the EVT file may be smaller than 1,048,608 bytes due to fewer events being observed. If the EVT file is smaller, then fewer total packets will be transferred.

# ASU Testing Command Set

The following table shows a set of commands that can be issued to the system to set the system parameters, take data, and transfer the data out.

These tables have a separate section for the MNS\_HV commands because the values that should be set are different depending on the hardware configuration.

In all cases, if there are fewer than four PMT modules hooked up to the analog board, then only the MNS\_HV and MNS\_TX commands for the modules which are attached need to be sent. The others can be omitted. For example, if there is a PMT module attached to channel 0 (anode 0 and dynode 0 on the analog board extension), then there is no need to send MNS\_HV to channel 1, 2, or 3. Similarly, in the same case, there is no need to request the 2DH data product for PMT 1, 2, or 3 because it will be only 0’s.

Table 2: Command Set for the ASU EM or Flight Unit

|  |  |
| --- | --- |
| Command | Description |
| MNS\_TRG\_0\_8850\n | Set trigger threshold to 8850 |
| MNS\_INT\_0\_-52\_88\_472\_6000\n | Set integration times |
| MNS\_ENABLE\_ACT\_0\n | Enable the 3.3 V |
| MNS\_HV\_0\_0\_bbb\n | Set high voltage potentiometer |
| MNS\_HV\_0\_1\_ddd\n | Set high voltage potentiometer |
| MNS\_ECAL\_0\_1.0\_0.0\n | Set energy calibration parameters |
| MNS\_DAQ\_0\_1 | Change system to data acquisition mode |
| MNS\_START\_0\_123456\_5 | Begin data acquisition; take data for 5 minutes |
| MNS\_END\_0\_654321 | End data acquisition |
| MNS\_TX\_0\_5\_1\_1\_0\_0 | Transfer CPS data product file |
| MNS\_TX\_0\_7\_1\_1\_0\_0 | Transfer EVT data product file |
| MNS\_TX\_0\_8\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 0 |
| MNS\_TX\_0\_11\_1\_1\_0\_0 | Transfer 2DH data product file for PMT 1 |
| Power Cycle |  |
| MNS\_WF\_0\_0\_5\_1 | Begin a Waveform Collection, request 5 waveforms |
| MNS\_TX\_0\_6\_1\_0\_0\_0 | Transfer WAV data product file |
| MNS\_DIR\_0\_0 | Report the contents of SD Card 0 |
| MNS\_DIR\_0\_1 | Report the contents of SD Card 1 |

To put together a useful MNS\_HV command, it must be known which channel(s) the module(s) are attached to. Once that is known, then the tap values should be taken, for a particular module, from Table 3.

Table 3: Module Numbers and their Associated HV Tap Values

|  |  |  |
| --- | --- | --- |
| Command | Description | HV Tap Value |
| MNS\_HV\_0\_0\_238 | FM.17 | 245 |
| MNS\_HV\_0\_1\_245 | FM.18 | 238 |
| MNS\_HV\_1\_0\_245 | FM.19 | 245 |
| MNS\_HV\_1\_1\_228 | FM.20 | 228 |

NOTE: If the system does not have an analog board attached, the MNS\_HV command will hang the system.

For the commands above, the following variables are identified:

* Detector Number = 0
* DAQ ID Number = 1
* DAQ Run Number = 1
* DAQ START real time = 123456
* DAQ Time out = 5 minutes
* DAQ END real time = 654321
* WAV Type = 0
* WAV Number of Waves = 5
* WAV ID number = 1

For the file transfers the end result should be:

* CPS: 3 packets
* EVT: 521 packets
* 2DH: 8 packets
* WAV: 42 packets
* DIR: variable

The files on the SD card should be the following sizes (for a 5 minute run):

* CPS: 4,556 bytes
* EVT: 1,048,608 bytes – for a set file
  + Each EVT data product is split into many set files, each file is 1 MB plus a header
  + The final EVT file will be smaller
* 2DH: 65,876 bytes – for each PMT

The file on the SD card from the waveform collection should be the following size (for 5 waveform buffers)

* WAV: 82,240 bytes

The files on the SD card should be the following sizes (for a 30 minute run):

* CPS: 25,444 bytes
* EVT: 1,048,608 bytes – for the first and intermediate set files
  + Each EVT data product is split into many set files, each file is 1 MB plus a header
* EVT = 228 < size < 1,048,608 bytes – for the final set file
* 2DH: 15,808 bytes – for each PMT

When transferring out the data products after a data acquisition run, the number of packets transferred per file transferred will be the following:

* CPS = 13 packets
* EVT = 529 packets
* 2DH = 34 packets

NOTE: if the event rate of the source is low, the EVT file may be smaller than 1,048,608 bytes due to fewer events being observed. If the EVT file is smaller, then fewer total packets will be transferred.

# Appendix

## APID Values for CCSDS Packets

The APID consists of the 3 LSBs from byte 4 and all of byte 5 in the CCSDS Packet. The 3 bytes in byte 4 are locked to being “010” to indicate that the detector number is detector 0. For this reason, we usually refer to a packet APID as just the value of byte 5, which is shown below in Table 2.

The APID field allows the Mini-NS to specify the type of data contained within the packet. The table below has the values for all possible Mini-NS APIDs and their data types.

Table 4: APID Vales for CCSDS Packets

|  |  |  |
| --- | --- | --- |
| APID LSB | Value | Description |
| Command Success | 0x00 | The input command was successfully executed |
| Command Failure | 0x11 | The input command failed somewhere during execution |
| Statement of Health | 0x22 | Packet contains statement of health information |
| LS Files Return | 0x33 | Packet contains filenames and file sizes for all the file on the Mini-NS SD card |
| Temperature | 0x44 | Packet with the most current temperatures. |
| MNS\_CPS | 0x55 | Neutron Counts per Second Data |
| MNS\_WAV | 0x66 | Waveform Data |
| MNS\_EVTS | 0x77 | Event by Event Data |
| MNS\_2DH | 0x88 | 2D Histogram Data |
| Log File | 0x99 | Log File |
| Configuration File | 0xAA | Configuration File |

## Commanded Functions That Return a Packet Other Than SUCCESS or FAILURE

Most commanded functions will return a CCSDS packet with the APID for success or failure when they return. The following functions may return a success or failure packet, but may return a packet with a different APID, as listed. For more information on these functions, see the Mini-NS ICD.

Table 5: Commanded Function Return Value Exceptions

|  |  |  |
| --- | --- | --- |
| Commanded Function | Possible Returns | APID |
| MNS\_DAQ | SUCCESS  FAILURE  DAQ\_SUCCESS | 0x00  0x11  0x00 + String in data bytes |
| MNS\_GETSTAT | SOH | 0x22 |
| MNS\_READTEMP | TEMP | 0x44 |
| MNS\_TX | CPS  WAV  EVTS  2DH  SUCCESS  FAILURE | 0x55  0x66  0x77  0x88  0x00  0x11 |