ASD Inc., a PANalytical company
2555 55th Street, Suite 100
Boulder, CO 80301

Phone: (303) 444-6522
Fax: (303) 444-6825
Email: nir.support@panalytical.com

TCPServer Developers Guide

Revision E

Table of Contents

Getting Started

This guide will provide an overview on how to install, configure and write a sample application to communicate with your ASD Ethernet instrument.

Network Configuration

To communicate through the Ethernet or Wireless interface, configure the host computer network adapter's Internet Protocol Version 4 (TCP/IPv4) to "Obtain an IP address automatically". The IP address for the ASD Instrument is set to 169.254.1.11.

What's New

Version 3.0

Integrate 802.11 n wireless interface.

Version 2.2

Integrate 802.11 g wireless interface.

Version 1.6

Add dark current floor check and update vnir drift values.

Version 1.5

Added AB Equal interface to A command.
New Interpolation routines.

Version 1.4

Added support for Trigger feedback.

Version 1.3

Added header structure to Acquire command
Added wireless capability

Version 1.2

Added ABORT command
Added IC command
Added V command
Added OPT command
Added support Vnir only instrument type.
Added support for Vnir/Swir1 instrument type.
Added support for Vnir/Swir2 instrument type.
Added support for Swir1/Swir2 instrument type.
Added support for Swir1 only instrument type.
Added support for Swir2 only instrument type.

Version 1.1

Released for Full Range instruments only.

Version 1.0

Initial Release

TCPServer API Documentation

The command interface is a comma delimited character string. The total number of parameters in the command structure is 4. An example command may look like the following: “A,1,10”. The first parameter is the command. Valid entries are defined in Table 1. The second parameter is the command type for the specified command. The third and fourth parameters in the command string are parameters for the command type. Valid entries are defined in Table 2. Table 3 defines the return structures of the requested command.

Table 1 Commands

Command	Description
A	Collect interpolated data.
ABORT	Aborts “A” and “OPT” commands
ERASE	Clears the contents of the flash.
IC	Instrument control command
INIT	Get, add or change ini file settings in the flash.
OPT	Optimize the instrument
RESTORE	Get and return the contents of the flash.
SAVE	Save ini file settings to the flash.
V	Version of firmware

Table 2 Command Type and Parameters

Param1	Param2	Param3	Param4	Description
A	*<None>*	*<None>*	*<None>*	Reset, then Acquire.
	1	1-32767	0-3	Set Sample Count. **Example:** “A,*1,10,0*” Sets the sample count to 10 with equal A and B scans.
	2	-1 - 15	*<None>*	Set Integration Time. Requires a third parameter: -1 - 15. This third parameter is the index value of the integration time. **Example:** “A,*2,0*” Sets the Vnir integration time to 17 ms.
	3	0-4096	0-4096	Set Gain and Offset of Swir1. Requires a third and fourth parameter. The third parameter is the Gain value to set. The fourth parameter is the Offset value to set. **Example:** “A,*3,500,2048*” Sets Swir1 Gain to 500 and Offset to 2048
	4	0-4096	0-4096	Set Gain and Offset of Swir2. Requires a third and fourth parameter. The third parameter is the Gain value to set. The fourth parameter is the Offset value to set. **Example:** “A,*4,500,2048*” Sets Swir2 Gain to 500 and Offset to 2048
	5	0-1	*<None>*	Toggle the shutter. Requires a third parameter. 0 to open the shutter. 1 to close the shutter. **Example:** “A,*5,0*” Open shutter. “A,*5,1*” Close shutter.
ABORT	<None>	<None>	*<None>*	Aborts current “A” and “OPT” command
ERASE	*<None>*	*<None>*	*<None>*	Clears the contents of the flash **Example:** “*ERASE*”
IC	**0 - 2**	0 - 4	-1 - 4096	Param2 values 0 – Swir1 1 – Swir2 2 – Vnir Param3 values 0 – Integration Time. Valid param4 values -1 - 15 1 – Gain Valid param4 values 0-4096 2 – Offset Valid param4 values 0-4096

				3 – Shutter Valid param4 values 0-1 4 – Trigger Valid param4 values 0 Param4 values – 0 - 4096 **Example:** “IC,2,0” Sets Vnir Integration Time to 17 ms “IC,0,1,500” Sets Swir1 Gain to 500 *“IC,1,2,2048”* Sets Swir2 Offset to 2048 *“IC,2,3,1”* Closes the Vnir shutter. *“IC,2,3,0”* Open the Vnir shutter.
INIT	**0**	30 char	***<None>***	Get value from flash. Requires a third parameter. The third parameter is the character string of a name of the value to get. ie. “SerialNumber” **Example:** “*INIT,0,SerialNumber*” gets the Serial Number from flash.
	1	30 char	double	Add a new to flash. Requires a third and fourth parameter. The third parameter is a character string of the name of the value ie. “SerialNumber. The fourth parameter is the value to set ie. “4012” **Example:** “*INIT,1,SerialNumber,4012*” Adds a Serial Number with a value of 4012 to the flash.
	2	30 char	double	Change a flash value. Requires a third and fourth parameter. The third parameter is a character string of the name of the value ie. “SerialNumber. The fourth parameter is the value to set ie. “4012” **Example:** “*INIT,2,SerialNumber,4028*” Changes the SerialNumber key to 4028.
OPT	**1**	***<None>***	***<None>***	Optimize VNIR device (BITMASK = 0x01). Upon successful completion of command, instrument values are set to optimized value(s).
	2	***<None>***	***<None>***	Optimize SWIR1 device (BITMASK = 0x02). Upon successful completion of command, instrument values are set to optimized value(s).
	3	***<None>***	***<None>***	Optimize VNIR and SWIR1 devices. Upon successful completion of command, instrument values are set to optimized value(s).
	4	***<None>***	***<None>***	Optimize SWIR2 device (BITMASK = 0x04). Upon successful completion of command, instrument values are set to optimized value(s).
	5	***<None>***	***<None>***	Optimize VNIR and SWIR2 device. Upon successful completion of command, instrument values are set to optimized value(s).
	6	***<None>***	***<None>***	Optimize SWIR1 and SWIR2 devices. Upon successful completion of command, instrument values are set to optimized value(s).
	7	***<None>***	***<None>***	Optimize VNIR, SWIR1 and SWIR2 devices. Upon successful completion of command, instrument values are set to optimized value(s).
RESTORE	**0 - 1**	***<None>***	***<None>***	Get and return the values from flash. Param2 0 - Loads the INI only 1 - Loads the INI and builds the calibration arrays. **Example:** “*RESTORE,1*”
SAVE	***<None>***	***<None>***	***<None>***	Save the current ini settings to flash. **Example:** “*SAVE*”
V	<None>	***<None>***	***<None>***	Returns the version of the TCP Server

Table 3 Return Packet structure.

Command	Return packet
A	// FRSpectrumHeader

```
struct Vnir_Header
{
    int IT;                              // Integration Time of vnir.
    int scans;                           // Number of scans in vnir region
    int max_channel;                     // Maximum DN value of vnir region
    int min_channel;                     // Minimum DN value of vnir region.
    int saturation;                      // Saturation Alarm   0 – no saturation  1 - saturation
    int shutter;                         // Shutter status   0 – Open   1 - Closed
    int drift;                           // Drift average value for defined drift channels
    int dark_subtracted;                 // Dark subtracted  0 – No    1 - Yes
    int reserved[8];
};

struct Swir_Header
{
   int tec_status;                       // Tec Alarm  0 – No Alarm  1 or 2 Alarm
   int tec_current;                      // DN value of TEC controller
   int max_channel;                      // Maximum DN value of swir region
   int min_channel;                      // Minimum DN value of swir region
   int saturation;                       // Saturation Alarm  0 – no saturation  1 - saturation
   int A_Scans;                          // Number of A Scans in swir region
   int B_Scans;                          // Number of B Scans in swir region
   int dark_current;                     // Averaged Dark Current value
   int gain;                             // gain value of swir region
   int offset;                           // offset value of swir region
   int scansize1;                        // A Scan - Number of channels before encoder index
                                         // B Scan – Number of channels after encoder index
   int scansize2;                        // A Scan - Number of channels after encoder index
                                         // B Scan – Number of channels before encoder index
   int dark_subtracted;                  // Dark subtracted  0 – No    1 - Yes
   int reserved[3];
};
struct SpectrumHeader
{
    int header;                          // Header code for Acquire
    int errbyte;                         // Error code for Acquire
    int sample_count;                    // Sample count of spectrum
    int trigger;                         // Trigger 0 – off   1 - on
    int voltage;                         // DN value of voltage.
    int current;                         // DN value of current.
    int temperature;                     // DN value of inside temperature.
    int motor_current;                   // DN value of motor current.
    int instrument_hours;                // Number of runtime hours since last calibration.
    int instrument_minutes;              // Number of runtime minutes since last calibration.
    int instrument_type;                 // 1 – 13  see version command for values
    int AB;                              // 0 – 3 see A command for value
    int reserved[4];
    Vnir_Header v_header;                // Vnir structure
    Swir_Header s1_header;               // Swir1 structure
    Swir_Header s2_header;               // Swir2 structure
};
//  Interpolated structure to return for Full Range Instrument
//  Applies to the FR_TCPServer firmware
//
struct FRInterpSpecStruct
{
    SpectrumHeader FRSpectrumHeader;  //256 bytes (64 words)
    float SpecBuffer [2151];
};
//
//  Interpolated structure to return for Vnir Spectrometers
// Applies to the V_TCPServer firmware
```

Command	Return packet

	//
	struct VInterpSpecStruct
	{
	SpectrumHeader VSpectrumHeader;
	float SpecBuffer [701];
	};//
	// Interpolated structure to return for Swir1 Swir2 Spectrometers
	// Applies to the S1S2_TCPServer firmware
	//
	struct S1S2InterpSpecStruct
	{
	SpectrumHeader S1S2SpectrumHeader;
	float SpecBuffer [1502];
	};//
	// Interpolated structure to return for Swir1 Spectrometers
	// Applies to the S1_TCPServer firmware
	//
	struct S1InterpSpecStruct
	{
	SpectrumHeader S1SpectrumHeader;
	float SpecBuffer [801];
	};
	//
	// Interpolated structure to return for Swir2 Spectrometers
	// Applies to the S2_TCPServer firmware
	//
	struct S2InterpSpecStruct
	{
	SpectrumHeader S2SpectrumHeader;
	float SpecBuffer [701];
	};
	};
	//
	// Interpolated structure to return for Vnir/Swir1 Spectrometers
	// Applies to the VS1_TCPServer firmware
	//
	struct VS1InterpSpecStruct
	{
	SpectrumHeader VS1SpectrumHeader;
	float SpecBuffer [1502];
	};
	//
	// Interpolated structure to return for Vnir/Swir2 Spectrometers
	// Applies to the VS2_TCPServer firmware
	//
	struct VS2InterpSpecStruct
	{
	SpectrumHeader VS2SpectrumHeader;
	float SpecBuffer [1402];
	};

ABORT	Struct ParamStruct
	{
	int header;
	int errbyte;
	char name[30];
	double value;
	int count;
	}

ERASE	struct InitStruct
	{
	int header; //header type used in TCP transfer.
	int errbyte; //error code
	char name [MAX_PARAMETERS][30]; //space for 200 entries with 30 character names
	double value [MAX_PARAMETERS]; //corresponding data values for the 200 entries
	int count; //The number of used entries

Command	Return packet

	int verify; //the checksum
	};

IC	struct InstrumentControlStruct
	{
	int header; // header type used in TCP transfer
	int errbyte; // error code
	int detector; // Detector number – 0 swir1, 1 swir2, 2 vnir
	int cmdType; // Command Type 0 IT, 1 Gain, 2 Offset, 3 Shutter, 4 Trigger
	int value; // Value issues 0 - 4096
	};

INIT	struct ParamStruct
	{
	int header; //header type used in TCP transfer.
	int errbyte; //error code
	char name [30]; //space for 200 entries with 30 character names
	double value; //corresponding data values for the 200 entries
	int count; //number of entries used
	}

OPT	struct OptimizeStruct
	{
	int header; //header type used in TCP transfer.
	int errbyte; //error code
	int itime; //optimized integration time
	int gain[2]; //optimized gain for 2 SWIRs
	int offset[2]; //optimized offset for 2 SWIRs
	};

RESTORE	struct InitStruct
	{
	int header; //header type used in TCP transfer.
	int errbyte; //error code
	char name [MAX_PARAMETERS][30]; //space for 200 entries with 30 character names
	double value [MAX_PARAMETERS]; //corresponding data values for the 200 entries
	int count; //The number of used entries
	int verify; //the checksum
	};

SAVE	struct InitStruct
	{
	int header; //header type used in TCP transfer.
	int errbyte; //error code
	char name [MAX_PARAMETERS][30]; //space for 200 entries with 30 character names
	double value [MAX_PARAMETERS]; //corresponding data values for the 200 entries
	int count; //The number of used entries
	int verify; //the checksum
	};.

V	struct VersionStruct
	{
	int header; // header type used in TCP transfer.
	int errbyte; // error code
	char version[30]; // 30 character Version and build
	double value; // Version number
	int type; // Type of instrument 1-Vnir, 4-Swir1, 5-Vnir/Swir1
	}; // 8-Siwr2, 9-Vnir/Swir2
	// 12-Swir1/Swir2, 13-Vnir/Swir1/Swir2

A – Acquire data

Description:

This command resets the detectors then collects and interpolates data at the current instrument settings.
Note: This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1

“A” Identifies Acquire command.

Param2

Not Used

Param3

Not Used

Param4

Not Used

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20
SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

“A”

Collects and interpolates data at the currently set sample count, integration time, gain and offsets.

A,1,*x*,*x* – Set sample count and Acquire data

Description:

This command sets the sample count, resets the detectors, collects and interpolates spectrum data.
Note: This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1

"A"	Identifies the Acquire command.

Param2

1	Set Sample Count command type.

Param3

1-32767	Sample count

Param4

0 – 3	Scan Type	0 – (Default) A and B Even spectrum averaging 1 – A only 2 – B only 3 – A and B.

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20

SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

“A,1,10”	Sets the sample count to 10 and returns interpolated data.

A,2,*x* – Set Integration time and Acquires data

Description:

This command sets the integration time, resets the detectors, collects and interpolates spectrum data.
Note: This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1

“A”	Identifies the Acquire command.

Param2

2	Set Integration Time command type.

Param3

Index	Integration Time
-1	8.5ms
0	17ms
1	34ms
2	68ms
3	136ms
4	272ms
5	544ms
6	1.09sec
7	2.18sec
8	4.35sec
9	8.70sec
10	17.41sec
11	34.82sec
12	1.16min
13	2.32min
14	4.64min
15	9.28min

Param4

Not Used

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20
SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

“A,2,0” Sets the integration time to 17ms.

A,3,*x*,*x* – Set Swir1 Gain and Offset and Acquires data

Description:

This command sets the gain and offset for swir1, resets the detectors, collects and interpolates spectrum data.
Note: This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1
“A” Identifies the Acquires command.

Param2
3 Set Gain and Offset for swir1 command type.

Param3
0- 4096 Gain value

Param4
0-4096 Offset value

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20
SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

"A,3,500,2048"	Sets the Gain of Swir1 to 500 and Offset to 2048.

A,4,*x*,*x* – Set Swir2 Gain and Offset and Acquires data

Description:

This command sets the gain and offset for swir2, resets the detectors, collects and interpolates spectrum data. *Note:* This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1

"A"	Identifies the Acquire command.

Param2

4	Set Gain and Offset for swir2 command type.

Param3

0- 4096	Gain value

Param4

0-4096	Offset value

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20
SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

"A,4,500,2048"	Sets the Gain of Swir2 to 500 and Offset to 2048.

A,5,*x* – Toggle the shutter and Acquires data

Description:

This command toggles the shutter for the vnir, resets the detectors, collects and interpolates spectrum data.
Note: This command requires the instrument ini and calibration arrays to be loaded into the flash. See RESTORE for Details.

Parameters

Param1

"A"	Identifies the Acquire command.

Param2

5	Toggle the shutter.

Param3

0	Open the shutter
1	Close the shutter

Param4

Not Used

Returns

```
Struct FRInterpSpecStruct
{
        SpectrumHeader FRSpectrumHeader;
        float SpecBuffer[2151];
}
```

header

H_NO_ERROR	100
H_COLLECT_ERROR	200
H_COLLECT_NOT_LOADED	300
H_RESET_ERROR	600
H_INTERPOLATE_ERROR	700

errbyte

NO_ERROR	0
NOT_READY	-1
NO_INDEX_MARKS	-2
TOO_MANY_ZEROS	-3
SCANSIZE_ERROR	-4
VNIR_TIMEOUT	-10
SWIR_TIMEOUT	-11
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
ABORT_ERROR	-18
VNIR_INTERP_ERROR	-20
SWIR1_INTERP_ERROR	-21
SWIR2_INTERP_ERROR	-22

SpecBuffer

Interpolated spectrum buffer.

See Table 3 for additional information on the return structures and header definition.

Example

"A,5,0"	Opens the Shutter
"A,5,1"	Closes the Shutter

ABORT – Abort command

Description:

This command Aborts the current “A” and “OPT” commands in the command queue.

Parameters

Param1

“ABORT” Identifies the Abort command.

Param2

Not Used.

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct ParamStruct
{
        int header;
        int errbyte;
        char name[30];
        double value;
        int count;
}
```

header

H_NO_ERROR 100

errbyte

NO_ERROR 0

name

“ABORT”

value

Not Used.

count

Not Used.

Example

“ABORT” Aborts the current “A” and “OPT” commands in the command queue.

ERASE – Clears the flash

Description:

This command clears the flash.

Parameters

Param1

“ERASE”	Identifies the ERASE command.

Param2

Not Used.

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct InitStruct
{
        int header;
        int errbyte;
        char name[200][30];
        double value[200];
        int count;
        int verify;
}
```

header

H_NO_ERROR	100
H_FLASH_ERROR	500

errbyte

NO_ERROR	0

name

Space for 200 entries with 30 character names.

value

Corresponding data value for 200 entries.

count

The number of used entries.

verify

The checksum value.

Example

“ERASE”	Clears the flash.

IC,0,1,*x* – Instrument Gain Control for SWIR1

Description:

This command sets the gain value for SWIR1.

Parameters

Param1

“IC”	Identifies the Instrument Control command.

Param2

0	SWIR1 Detector

Param3

1	Gain control

Param4

0-4096	Gain value to set

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
2	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

0 - 4096

Example

“IC,0,1,500”	Sets the Gain to 500 for SWIR1.

IC,0,2,*x* – Instrument Offset Control for SWIR1

Description:

This command sets the offset value for SWIR1.

Parameters

Param1

"IC"	Identifies the Instrument Control command.

Param2

0	SWIR1 Detector

Param3

2	Offset control

Param4

0-4096	Offset value to set

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
3	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

0 - 4096

Example

“IC,0,2,2048”	Sets the Offset to 2048 for SWIR1.

IC,1,1,*x* – Instrument Gain Control for SWIR2

Description:

This command sets the gain value for SWIR2.

Parameters

Param1

"IC"	Identifies the Instrument Control command.

Param2

1	SWIR2 Detector

Param3

1	Gain control

Param4

0-4096	Gain value to set

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
2	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

0 - 4096

Example

“IC,1,1,500”	Sets the Gain to 500 for SWIR2.

IC,1,2,*x* – Instrument Offset Control for SWIR2

Description:

This command sets the offset value for SWIR2.

Parameters

Param1

"IC"	Identifies the Instrument Control command.

Param2

1	SWIR2 Detector

Param3

2	Offset control

Param4

0-4096	Offset value to set

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
2	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

0 - 4096

Example

“IC,1,2,2048”	Sets the Offset to 2048 for SWIR2.

IC,2,0,*x* – Instrument Integration Time Control for VNIR

Description:

This command sets the integration time value index for VNIR.

Parameters

Param1

“IC”	Identifies the Instrument Control command.

Param2

2	VNIR Detector

Param3

0	Integration Time control

Param4

Index	Integration Time
-1	8.5ms
0	17ms
1	34ms
2	68ms
3	136ms
4	272ms
5	544ms
6	1.09sec
7	2.18sec
8	4.35sec
9	8.70sec
10	17.41sec
11	34.82sec
12	1.16min
13	2.32min
14	4.64min
15	9.28min

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0

NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
2	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

-1 - 15

Example

“IC,2,0,0” Sets the integration time index to 17ms for the VNIR detector.

IC,2,3,*x* – Instrument Shutter Control for VNIR

Description:

This command toggles the shutter for VNIR.

Parameters

Param1

"IC"	Identifies the Instrument Control command.

Param2

2	VNIR Detector

Param3

3	Shutter control command

Param4

0	Open shutter
1	Close shutter

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
NOT_READY	-1
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
PARAM_ERROR	-19

detector

0	SWIR1
1	SWIR2
2	VNIR

cmdType

0	Integration Time
1	Gain
2	Offset
3	Shutter

values

0 - 4096

Example

“IC,2,3,0”	Opens the shutter for the VNIR detector.
“IC,2,3,1”	Closes the shutter for the VNIR detector.

IC,2,4,0 – Instrument Trigger Reset

Description:

This command resets the Trigger for activation. When the trigger is pressed, the LEDs turn on and the instrument sends a “Trigger” character string to the client. The trigger becomes inactive until it has been reset. Use this command to turn off the LEDs and reactivate the trigger.

Parameters

Param1

“IC”	Identifies the Instrument Control command.

Param2

2	VNIR Detector

Param3

4	Trigger Reset command

Param4

0	Reset

Returns

```
Struct InstrumentControlStruct
{
        int header;
        int errbyte;
        int detector;
        int cmdType;
        int value;
}
```

header

H_NO_ERROR	100
H_INSTRUMENT_CONTROL_ERROR	900

errbyte

NO_ERROR	0
PARAM_ERROR	-19

detector

2	Vnir

cmdType

4	Trigger Reset

values

0 -	Reset

Example

“IC,2,4,0”	Resets the Trigger by turning off the LEDs and resetting the register.

INIT,0,*x* – Gets parameter from flash

Description:

This command gets a parameter stored in flash.
Note: This command requires a RESTORE command to have been called prior to retrieving the parameter values.

Parameters

Param1

"INIT"	Identifies the INIT command.

Param2

0	Gets a parameter from flash.

Param3

30 chars	Parameter name. See RESTORE command for possible names.

Param4

Not Used

Returns

```
Struct ParamStruct
{
        int header;
        int errbyte;
        char name[30];
        double value;
        int count;
}
```

header

H_NO_ERROR	100
H_INIT_ERROR	400

errbyte

NO_ERROR	0
MISSING_PARAMETER	-8

name

Name of parameter up to 30 character long.

value

Corresponding data value for parameter.

count

The number of used entries.

Example

"INIT,0,SerialNumber"	Returns the Serial Number stored in Flash.

INIT,1,*x*,*x* – Adds a parameter to flash

Description:

This command adds a parameter to be stored in flash.
Note: This command requires the Save command to permanently store the value in flash.

Parameters

Param1

“INIT”	Identifies the INIT command.

Param2

1	Adds a parameter to flash.

Param3

30 chars	Parameter name

Param4

Double	Value of the Parameter

Returns

```
Struct ParamStruct
{
        int header;
        int errbyte;
        char name[30];
        double value;
        int count;
}
```

header

H_NO_ERROR	100
H_INIT_ERROR	400

errbyte

NO_ERROR	0
INI_FULL	-7

name

Name of parameter up to 30 character long.

value

Corresponding data value for parameter.

count

The number of used entries.

Example

“INIT,1,SerialNumber,4012”	Adds the SerialNumber parameter with a value of 4012 to Flash.

INIT,2,*x*,*x* – Changes a parameter stored in flash

Description:

This command changes a parameter stored in flash.
Note: This command requires a RESTORE command to have been called prior to changing the parameter values. This command also requires the Save command to permanently store the value in flash.

Parameters

Param1

"INIT"	Identifies the INIT command.

Param2

2	Changes a parameter in flash.

Param3

30 chars	Parameter name. See RESTORE command for possible names

Param4

Double	Value of the Parameter

Returns

```
Struct ParamStruct
{
        int header;
        int errbyte;
        char name[30];
        double value;
        int count;
}
```

header

H_NO_ERROR	100
H_INIT_ERROR	400

errbyte

NO_ERROR	0
MISSING_PARAMETER	-8

name

Name of parameter up to 30 character long.

value

Corresponding data value for parameter.

count

The number of used entries.

Example

"INIT,1,SerialNumber,6027"	Changes the SerialNumber parameter to 6027 in Flash.

OPT,1 – Optimize VNIR detector

Description:

This command optimizes the VNIR detector.

Parameters

Param1

"OPT"	Identifies the OPT command.

Param2

1	VNIR detector	(BITMASK = 0x01)

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,1”	Optimize VNIR detector.

OPT,2 – Optimize SWIR1 detector

Description:

This command optimizes the SWIR1 detector.

Parameters

Param1

“OPT” Identifies the OPT command.

Param2

2 SWIR1 detector (BITMASK = 0x02)

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,2”	Optimize SWIR1 detector.

OPT,3 – Optimize VNIR and SWIR1 detectors

Description:

This command optimizes the VNIR and SWIR1 detectors.

Parameters

Param1

“OPT”	Identifies the OPT command.

Param2

3	VNIR and SWIR1 detector

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,3” Optimize VNIR and SWIR1 detectors.

OPT,4 – Optimize SWIR2 detector

Description:

This command optimizes the SWIR2 detector.

Parameters

Param1

"OPT" Identifies the OPT command.

Param2

4	SWIR2 detector	(BITMASK=0x04)

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,4”	Optimize VNIR and SWIR1 detectors.

OPT,5 – Optimize VNIR and SWIR2 detectors

Description:

This command optimizes the VNIR and SWIR2 detectors.

Parameters

Param1

“OPT”	Identifies the OPT command.

Param2

5	VNIR and SWIR2 detector

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,5”	Optimize VNIR and SWIR2 detectors.

OPT,6 – Optimize SWIR1 and SWIR2 detectors

Description:

This command optimizes the SWIR1 and SWIR2 detectors.

Parameters

Param1
"OPT" Identifies the OPT command.

Param2
6 SWIR1 and SWIR2 detector

Param3
Not Used.

Param4
Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,6” Optimize SWIR1 and SWIR2 detectors.

OPT,7 – Optimize VNIR, SWIR1 and SWIR2 detectors

Description:

This command optimizes the VNIR, SWIR1 and SWIR2 detectors.

Parameters

Param1

“OPT”	Identifies the OPT command.

Param2

7	VNIR, SWIR1 and SWIR2 detector

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct OptimizeStruct
{
        int header;
        int errbyte;
        int itime
        int gain[2]
        int offset[2]
}
```

header

H_NO_ERROR	100
H_OPTIMIZE_ERROR	800

errbyte

NO_ERROR	0
NOT_READY	-1
MISSING_PARAMETER	-8
VNIR_NOT_READY	-12
SWIR1_NOT_READY	-13
SWIR2_NOT_READY	-14
VNIR_OPT_ERROR	-15
SWIR1_OPT_ERROR	-16
SWIR2_OPT_ERROR	-17
ABORT_ERROR	-18

itime

-1	Error if gain and offset are -1
-1 - 15	Integration time for the VNIR detector.

gain

-1	Error
[1] 0 – 4096	gain value for first SWIR detector.
[2] 0 – 4096	gain value for second SWIR detector.

offset

-1	Error
[1] 0 – 4096	offset value for first SWIR detector.
[2] 0 – 4096	offset value for second SWIR detector.

Example

“OPT,7” Optimize VNIR, SWIR1 and SWIR2 detectors.

RESTORE,x – Loads the flash into RAM

Description:

This command loads the values stored in flash into RAM. In version 1.5, this command takes upwards to 10 seconds to complete.
Note: "RESTORE,1" is required for 1.5 version and greater for Acquire (A) command to work properly.

Parameters

Param1

"RESTORE"	Identifies the RESTORE command.

Param2

0	Restores INI only
1	Restores INI and build calibration Arrays.

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct InitStruct
{
        int header;
        int errbyte;
        char name[200][30];
        double value[200];
        int count;
        int verify;
}
```

header

H_NO_ERROR	100
H_INIT_ERROR	400

errbyte

NO_ERROR	0
INSTRUMENT_INI_LOAD_ERROR	-1
VNIR_INI_LOAD_ERROR	-2
SWIR1_INI_LOAD_ERROR	-3
SWIR2_INI_LOAD_ERROR	-4

name

Space for 200 entries with 30 character names.

INI entries below

Version
SerialNumber
CalibrationNumber
InstrumentType

Detectors
StartingWavelength
EndingWavelength
InstrumentType
InstrumentHours
InstrumentMinutes
ConnectionIdleTimeout
ConnectionOverrideTimeout
OptType
OptimizationLogEnabled
OptimizationTimeOutSeconds
EnableTrigger
MotorCurrentAdjustment
MotorCurrentThreshold
BoardAssemblyVersion
VDetectorType
VRealChannels
VStartingWavelength
VEndingWavelength
VUseLinear
VCalWavelengthStart
VCalWavelengthStep
VCalStartingWavelengthBlockV
VCalWavelengthStepBlockV
VDeltaStepBlockV
VDeltaSquareStepBlockV
VDriftChannelStart
VDriftChannelCount
VStartingIntegrationTimeIndex
VMinIntegrationTimeIndex
VMaxIntegrationTimeIndex
VDarkCurrentCorrection
VDarkSampleCount
VInterpolate
VVertex
S1DetectorType
S1RealChannels
S1StartingWavelength
S1EndingWavelength
S1IndexChannel
S1DarkStart
S1DarkSize
S1AdjustOffset
S1CalStartingWavelengthBlockA
S1CalWavelengthStepBlockA
S1DeltaStepBlockA
S1DeltaSquareStepBlockA
S1CalStartingWavelengthBlockB
S1CalWavelengthStepBlockB
S1DeltaStepBlockB
S1DeltaSquareStepBlockB
S1Interpolate
S1Vertex
S2DetectorType

S2RealChannels
S2StartingWavelength
S2EndingWavelength
S2IndexChannel
S2DarkStart
S2DarkSize
S2AdjustOffset
S2CalStartingWavelengthBlockA
S2CalWavelengthStepBlockA
S2DeltaStepBlockA
S2DeltaSquareStepBlockA
S2CalStartingWavelengthBlockB
S2CalWavelengthStepBlockB
S2DeltaStepBlockB
S2DeltaSquareStepBlockB
S2Interpolate
S2Vertex

value

Corresponding data value for 200 entries.

count

The number of used entries.

verify

The checksum value.

Example

“RESTORE,1” Loads the flash into RAM and builds calibration arrays.

SAVE – Saves the values in RAM to flash

Description:

This command saves the parameters in RAM to flash.

Parameters

Param1

“SAVE” Identifies the SAVE command.

Param2

Not Used.

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct InitStruct
{
        int header;
        int errbyte;
        char name[200][30];
        double value[200];
        int count;
        int verify;
}
```

header

H_NO_ERROR	100
H_FLASH_ERROR	500

errbyte

NO_ERROR	0

name

Space for 200 entries with 30 character names.

value

Corresponding data value for 200 entries.

count

The number of used entries.

verify

The checksum value.

Example

“SAVE” Saves the parameters in RAM to flash.

V – Version

Description:

This command returns the version of the firmware.

Parameters

Param1

“V” Identifies the Version command.

Param2

Not Used.

Param3

Not Used.

Param4

Not Used.

Returns

```
Struct ParamStruct
{
        int header;
        int errbyte;
        char name[30];
        double value;
        int type;
}
```

header

H_NO_ERROR 100

errbyte

NO_ERROR 0

name

Version of the firmware.

value

Version value.

type

Type of instrument

VNIR	1
SWIR1	4
VNIR/SWIR1	5
SWIR2	8
VNIR/SWIR2	9
SWIR1/SWIR2	12
VNIR/SWIR1/SWIR2	13

Example

“V” Returns the Version of the firmware.

Dark Current Collection

Dark Current collection is the process of blocking light coming into the instrument, then collecting the internal generated signal so that it can be subtracted from the external signal. Blocking the incoming light into the instrument can be accomplished with a mechanical shutter or by capping the fiber. A more efficient way of collecting dark current is through a dark current look up table. Recent testing has shown the dark current in the VNIR region to be stable. This stability allows for the use of a table to record the dark current values. The dark current table is easily generated with the Dark Current Calibration (DCC) utility supplied as part of the software package. Use of the table improves data collection rates by eliminating the time needed for the mechanical shutter process. Any changes in the dark current values due to normal fluctuations are small and are automatically adjusted by the software's Drift Lock feature. The use of the dark current table will be the default configuration on new instruments and can also be retroactively applied to existing Ethernet instruments.

The following is the Dark Correction algorithm:

$$\forall i \in \{0, \dots, n\}\ DC_S(i) = T_s(i) - D_s(i) + (V_{DarkCurrentCorrection} + (T_{drift} - D_{drift}))$$

Where:
n = size of the VNIR spectrum
DC_S = dark corrected spectrum
T_S = current measured spectrum
D_S = dark measured spectrum
$V_{DarkCurrentCorrection}$ = dark current correction constant
T_{drift} = current measured drift value
D_{drift} = dark measured drift value

The following describes the Dark Current Collection process for the three different methods:

a. Has Shutter
b. Has Dark File
c. No Shutter or Dark File

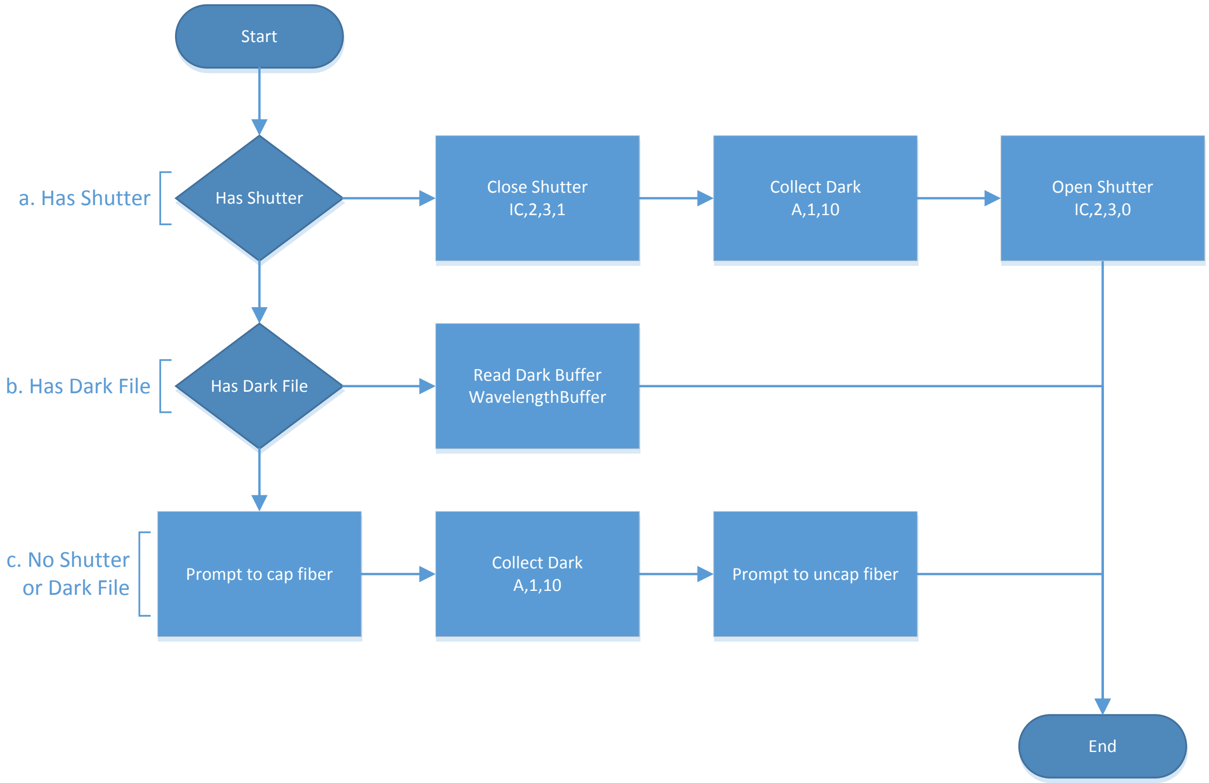

Figure 1: Dark Current Collection Process

1. Block Incoming Light
   a. Has Shutter
   Close Shutter – IC,2,3,1
   b. Has Dark File
   Open Dark Current ini file. This is will be in the form <*serial number*>_<*calibration number*>_DarkCurrent.ini (ie. 18343_2_DarkCurrent.ini). Where <*serial number*> is the serial number of the instrument and <*calibration number*> is the calibration number for the instrument.
   c. No Shutter or Dark File
   Prompt to cap the fiber.

2. Collect Dark Measured Spectrum - D_S
   a. Has Shutter
   Acquire spectrum from instrument – A,1,10
   b. Has Dark File
   Read the *WavelengthBuffer* from dark current file where the *Index* matches the current Integration Time. The look up table consists of channel data and wavelength data for each integration time.
   c. No Shutter or Dark File
   Acquire spectrum from instrument – A,1,10.

3. Read Dark Drift of Dark Measured Spectrum - D_{drift}
   a. Has Shutter

Read the *drift* value from Vnir Header.

b. Has Dark File
Read the *drift* value from dark current file where the Index matches the current Integration Time.

c. No Shutter or Dark File
Read the *drift* value from Vnir Header.

4. Collect Current Measured Spectrum - T_S

a. Has Shutter
Acquire spectrum from instrument – A,1,10.

b. Has Dark File
Acquire spectrum from instrument – A,1,10.

c. No Shutter or Dark File
Acquire spectrum from instrument – A,1,10.

5. Read Dark Drift of Current Measured Spectrum - T_{drift}

a. Has Shutter
Read the *drift* value from Vnir Header

b. Has Dark File
Read the *drift* value from dark current file where the Index matches the current Integration Time.

c. No Shutter or Dark File
Read the *drift* value from Vnir Header.

6. Compute Dark Corrected Spectrum - DC_S

Note: VNIR DarkCurrentCorrection constant, VNIR StartingWavelength and EndingWavelength can be obtained from the Instrument using the INIT command.

VNIR StartingWavelength

$\mathrm{V_{StartingWavelngth}} = \mathrm{INIT, 0, VStartingWavelength}$

VNIR EndingWavelength

$V_{EndingWavelngth} = INIT, 0, VEndingWavelength$

VNIR DarkCurrentCorrection constant

$V_{DarkCurrentCorrection} = INIT, 0, VDarkCurrentCorrection$

Loop through the VNIR spectrum, subtract the dark spectrum from the current spectrum and add the Drift correction.

```
$for(int\ i = 0; i <\ V_{EndingWavelength}\ - V_{StartingWavelength}; i++)$
{
  $DC_S(i) = T_S(i) - D_s(i) + \left(V_{DarkCurrentCorrection} + \left(T_{drift} - D_{drift}\right)\right)$
}
```

Writing a TCP Client

A TCP Client application is required to initiate a connection and issue commands to the TCP Server. A sample application has been provided to demonstrate the topics below. The sample application is located under the samples folder.

Making and closing a connection

To connect to a TCP Server, the TCP Client application must know the IP Address and Port number of the TCP Server. Please refer to the *Determine the network configuration* section for setting the TCP Server's IP Address. The ASD Instrument's IP address is 169.254.1.11. The Port number is 8080.

Connecting

The following code snippet shows how to make a connection to a TCP server with an address of 169.254.1.11 on port 8080.

```
//
// Initialize WSA
//
if(WSAStartup(MAKEWORD(2,2), &WsaDat)!=0)
{
        printf("WSA Initialization failed.");
        return;
}
//
// Create Socket
//
Socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP );

if(Socket == INVALID_SOCKET)
{
        printf("Socket creation failed.");
}
//
// Connect to TCP Server
//
SOCKADDR_IN SockAddr;

SockAddr.sin_port = htons(8080);
SockAddr.sin_family = AF_INET;
SockAddr.sin_addr.S_un.S_addr = inet_addr("169.254.1.11");


int RetVal = connect(Socket, (SOCKADDR *)(&SockAddr), sizeof(SockAddr));
if(RetVal != 0)
{
        int l = WSAGetLastError();
        printf("Failed to establish connection with server. %d\n", l);

}
```

Closing the Connection

```
        //
        // Close the Socket
        //
        closesocket(Socket);

        //
        // Clean of the Winsock library
        //
        WSACleanup();
```

The following code snippet shows how to disconnect from the TCP Server.

Reading the starting and ending wavelength

Before reading the starting and ending wavelength of the TCP Server, the instrument’s INI must be loaded into flash. Each instrument comes with the INI pre loaded. To update the instrument’s INI, please refer to the Net Configuration Guide. Reading the instrument’s starting and ending wavelength uses the INIT,0,x command. The following code snippet demonstrates reading the starting and ending wavelength.

Starting Wavelength

```
CString  strCommand = “INIT,0,StartingWavelength”);

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );
```

Ending Wavelength

```
CString  strCommand = “INIT,0,EndingWavelength”);

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );
```

Optimize

The following code snippet demonstrates how to optimize the instrument.

```
CString strCommand = “OPT,7”;

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );
```

Acquiring data

The following code snippet demonstrates how to Acquire data from the instrument.

```
//
// Initialize the FR Spectrum Structure
//
```

```
FRInterpSpecStruct *iss;

iss = (FRInterpSpecStruct *)malloc(sizeof(*iss));
//
// Collect 10 samples
//
CString strCommand = “A,1,10”;

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );

//
// Loop until the data has been collected
//
int bytesRecv = 0;
char *recvbuf = new char[bytesToRecv];
totalBytesRecv = 0;

while( totalBytesRecv < bytesToRecv)
{
        bytesRecv = recv( Socket, recvbuf, bytesToRecv, 0 );
        if (bytesRecv == SOCKET_ERROR)
                break;

        if ( bytesRecv == 0 || bytesRecv == WSAECONNRESET )
        {
                printf( "Connection Closed.\n");
                break;
        }
        printf( "Bytes Recv: %ld\n", bytesRecv );

        memmove(&recvBuf[totalBytesRecv], recvbuf, bytesRecv);
        totalBytesRecv += bytesRecv;
}

//
// Convert the Header and errbyte from big endian to little endian to see if it is good data
//
iss->FRHeader.Header = ntohl(iss->FRHeader.Header);
iss->FRHeader.errbyte = ntohl(iss->FRHeader.errbyte);

if(iss->FRHeader.Header == 100)
{
        unsigned long z;
        //
        // Convert the buffer from big endian to little endian and store the value as a float
        //
        for(int i=0;i<(sizeof(iss->SpecBuffer) / sizeof(float));i++)
        {
                z = ntohl(iss->SpecBuffer[i].i);
                memcpy(&iss->SpecBuffer[i].f,&z,sizeof(float));
        }

}
```

Displaying a Dark Corrected Spectrum

The following code snippet demonstrates how to display a dark corrected spectrum using a shutter.

```
//
// Close the shutter
//
CString strCommand = "IC,2,3,1");

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );
//
// Initialize the FR Dark Spectrum Structure
//
FRInterpSpecStruct *issDarkSpectrum;

issDarkSpectrum = (FRInterpSpecStruct *)malloc(sizeof(*issDarkSpectrum));
//
// Collect 10 Dark Samples
//
CString strCommand = “A,1,10”;

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );

//
// Convert the received data to float
//
……  Code omitted for brevity – See Acquire section for details
//
// Assign Dark drift value
dark_drift = issDarkSpectrum.FRHeader.v_header.drift;
//
// Open the shutter
//
strCommand = "IC,2,3,0");

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );
//
// Initialize the FR Spectrum Structure
//
FRInterpSpecStruct *iss;

iss = (FRInterpSpecStruct *)malloc(sizeof(*iss));

//
// Acquire data to subtract the dark
//
strCommand = “A,1,10”;

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );

//
// Convert the received data to float
//
……  Code omitted for brevity – See Acquire section for details
```

```
//
// Assign Current drift value
current_drift = iss.FRHeader.v_header.drift;

//
// Subtract the Dark Spectrum from the current spectrum
//
if(iss->FRHeader.Header == 100)
{
        // Compute drift
        float drift = m_iVnirDarkCurrentCorrection + (current_drift – dark_drift);
        // Subtract dark
        for(int i = 0; i < ((m_iVnirEndingWavelength + 1) - m_iStartingWavelength); i++)
                iss->SpecBuffer[i].f -= issDarkSpectrum->SpecBuffer[i].f + drift;
}
```

Displaying a Reflectance Spectrum

The following code snippet demonstrates how to display a reflectance spectrum.

```
//
// Collect and store a reference spectrum
//

//
// Initialize the Reference FR Spectrum Structure
//
FRInterpSpecStruct *issReference;

issReference = (FRInterpSpecStruct *)malloc(sizeof(*issReference));

CString strCommand = “A,1,10”;

bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );

//
// Convert the received data to float
//
……  Code omitted for brevity – See Acquire section for details
//
//
// Collect a current Spectrum to compute reflectance
//
//
// Initialize the FR Spectrum Structure
//
FRInterpSpecStruct *iss;

iss = (FRInterpSpecStruct *)malloc(sizeof(*iss));

//
// Acquire current data
//
strCommand = “A,1,10”;
```

```
bytesSent = send( Socket, strCommand, strCommand.GetLength(), 0 );

//
// Convert the received data to float
//
……  Code omitted for brevity – See Acquire section for details
//
//
// Compute reflectance
//
if(iss->FRHeader.Header == 100)

{

        // Compute Reflectance

        for(int i = 0; i < ((m_iEndingWavelength + 1) - m_iStartingWavelength); i++)

            iss->SpecBuffer[i].f = iss->SpecBuffer[i].f/ issReference->SpecBuffer[i].f;

}
```

Normalizing a Spectrum

The following code snippet demonstrates how to normalize spectrum.

```
//
// Acquire data – see the Acquire section
//
// Create the Normalized structure
//
FRInterpSpecStruct *issNormalize;
issNormalize = (FRInterpSpecStruct*)malloc(sizeof(*issNormalize));

if(iss->Header == 100)
{

        int i;
        // Normalize Vnir to IT-17ms
        for(i = 0; i < ((m_iVnirEndingWavelength + 1) - m_iStartingWavelength); i++)
                issNormalize->SpecBuffer[i].f = iss->SpecBuffer[i].f/ (1<<it);

        // Normalize Swir1 Gain to 4096
        float gc = 256;
        float n = s1g/gc;
        for(i = (m_iVnirEndingWavelength + 1) - m_iStartingWavelength;
                i < ((m_iSwir1EndingWavelength + 1) - m_iStartingWavelength); i++)
                issNormalize->SpecBuffer[i].f = iss->SpecBuffer[i].f * n;
        // Normalize  Swir2 Gain to 4096
        n = s2g/gc;
        for(i = (m_iSwir1EndingWavelength + 1) - m_iStartingWavelength;
                i < ((m_iSwir2EndingWavelength + 1) - m_iStartingWavelength); i++)
                issNormalize->SpecBuffer[i].f = iss->SpecBuffer[i].f * n ;

}
```

Support

ASD Inc. a PANalytical company
2555 55th Street, Suite 100
Boulder, CO 80301

Phone: 303-444-6522
Fax: 303-444-6825
Web site: www.asdi.com
Email: nir.support@panalytical.com