NeoSpectra Micro Developers' Guide: Electrical interface requirements

Manual notes

About this manual
This manual will give you a complete overview about the electrical interface of Neospectra Micro. This includes understanding the function of the different NeoSpectra Micro's electronic components, pins assignment, electrical characteristics, power network and power modes of the module as well as the details of the communications interface.

Who should read this manual?
Teams who will be working on the development of the end-use application system and the hardware implementation/integration of the host system.

Relevant products

NeoSpectra Micro

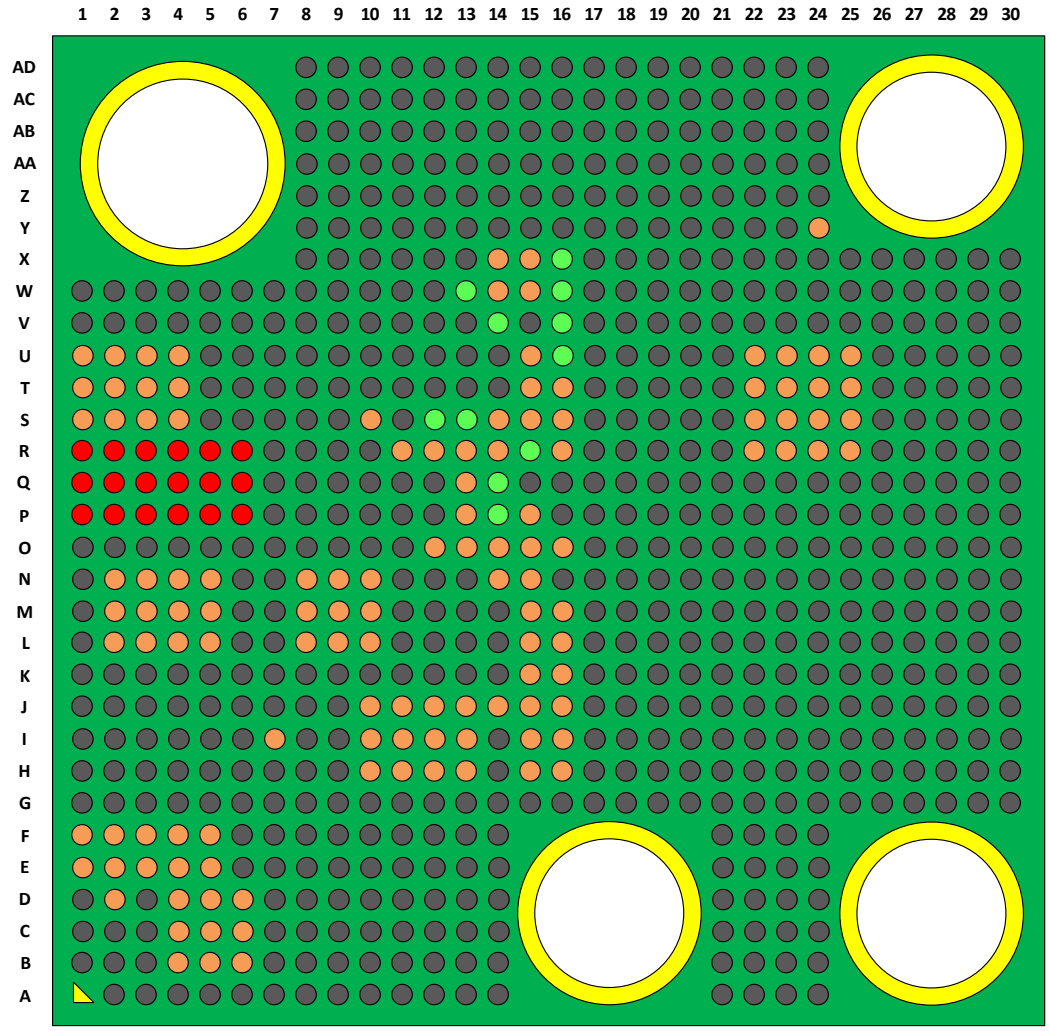
Tags:

Pins assignment, electrical characteristics, power network, power modes, SPI communication, modes of operation, operation guide

Table of Contents

1. Pin Assignment

NeoSpectra Micro features a ball grid array (BGA) interface that makes it ready to be integrated in larger systems with a compact size. The interface consists of 30x30 ball grid array; each has a 600um diameter and the pitch between balls is 1mm. A notch resides at the bottom left corner indicating the location of the first ball and the proper orientation of the module. **Figure 1** shows the top view of the BGA of NeoSpectra Micro with a color coded map that specifies the pin assignment of the different balls. The exact function of each pin is shown in **Table 1**.



Top View
31 mm X 31 mm
30 X 30 Balls Grid Array
600 um ball diameter
1 mm pitch

3.3V Supply Pins	
Ground Pins	
3.3V Digital Control Pins	
P14	SPI_CLK
Q14	WKUP
R15	EXTRG
S12	SPI_MOSI
S13	SPI_MISO
V14	DRDY
V16,U16	EN
W13	SPI_CSB
W16	INTRPT
X16	SPI_MODSEL
Not Connected	

Figure 1 - Top view of NeoSpectra Micro BGA

Table 1 - Pin Description

Pin	Function	Direction
VDD_3V	3.3V Power Supply	Supply
GND	Power Supply Ground	Supply
EN	Module enable	In
SPI_CSB	SPI chip select	In
SPI_CLK	SPI clock	In
SPI_MOSI	SPI master output/ slave input	In
SPI_MISO	SPI master input/ slave output	Out
DRDY	Data ready – Indicates module is ready to execute user commands	Out
INTRPT	Interrupt – Indicates an error/warning in the latest executed operation	Out
WKUP	Module Wakeup – Used to wake the module up from sleep mode	In
EXTRG	External Trigger – Initiates a new PSD acquisition operation	In
SPI_MODSEL	SPI Mode Select – Selects the operating SPI mode of NeoSpectra Micro; '0': Normal mode, '1': High speed mode	Out

2. Power-up Sequence

1- The user has to make sure that all the module interface pins are kept low as long as the module is powered-off so as to avoid any unexpected behavior.
2- To power-on the module, the 3.3V power supply should be provided, then the power management circuit should be enabled by setting the EN pin to '1'.
3- After 25ms, user starts to check for 'DRDY' pin and wait it to be '1'; this indicates that NeoSpectra Micro is ON and ready to execute user commands.
4- NeoSpectra Micro is now ready. To start SPI communication, user has to check first for the SPI_MODSEL pin to select the proper SPI speed mode as described in Section 6.1 "SPI Slave Physical Layer".
5- After selecting the SPI speed mode, user can write the needed configurations and execute the different operations through the SPI interface as described later in Section 6.3 "Operations Guide".

3. Power Modes

3.1. Overview

NeoSpectra Micro provides a power management scheme featuring different low-power modes to satisfy the needs of low-power/battery-operated applications.

Four different power modes are provided:

1. Active mode.
2. Stand-by mode.
3. Sleep mode.
4. Power-off mode.

Power modes vary in power consumption to balance the speed and the power requirements of the different applications. The more power reduction the mode gives, the more time it needs to wake up.

3.2. Active Mode

Active mode is the normal operating mode. NeoSpectra Micro runs the different components to serve the user command in the shortest time with the best SNR while optimizing the power consumption as possible.

NeoSpectra Micro enters active mode upon receiving any command during stand-by mode. The typical current consumption of the active mode is 1250 mA while all components are running.

3.3. Stand-by Mode

Stand-by mode is the default power mode for NeoSpectra Micro. Module enters stand-by mode automatically after powering up and after finishing execution of the different user commands. In stand-by mode, NeoSpectra Micro module is ready to execute user commands or stream out the different output data of the previous operations.

NeoSpectra Micro leaves the stand-by mode upon a command from the user either to execute a certain operation (Active Mode) or to enter any different low power modes. The typical current consumption of the stand-by mode is 70 mA.

3.4. Sleep Mode

Sleep mode is a kind of low-power mode. It is designed to preserve all the user data without the need of flash programming in addition to a faster wakeup time at the cost of a bit higher power consumption.

Entering sleep mode doesn't power off the internal RAMs so the user can enter sleep mode without losing whatever data. NeoSpectra Micro enters sleep mode with SPI user command and exits with setting WKUP pin for a minimum period of 1 ms. Stand-by to sleep mode transition time is 1 ms (maximum) while the sleep to stand-by transition time is 2.5 ms (maximum). The typical current consumption of the sleep mode is 31 mA.

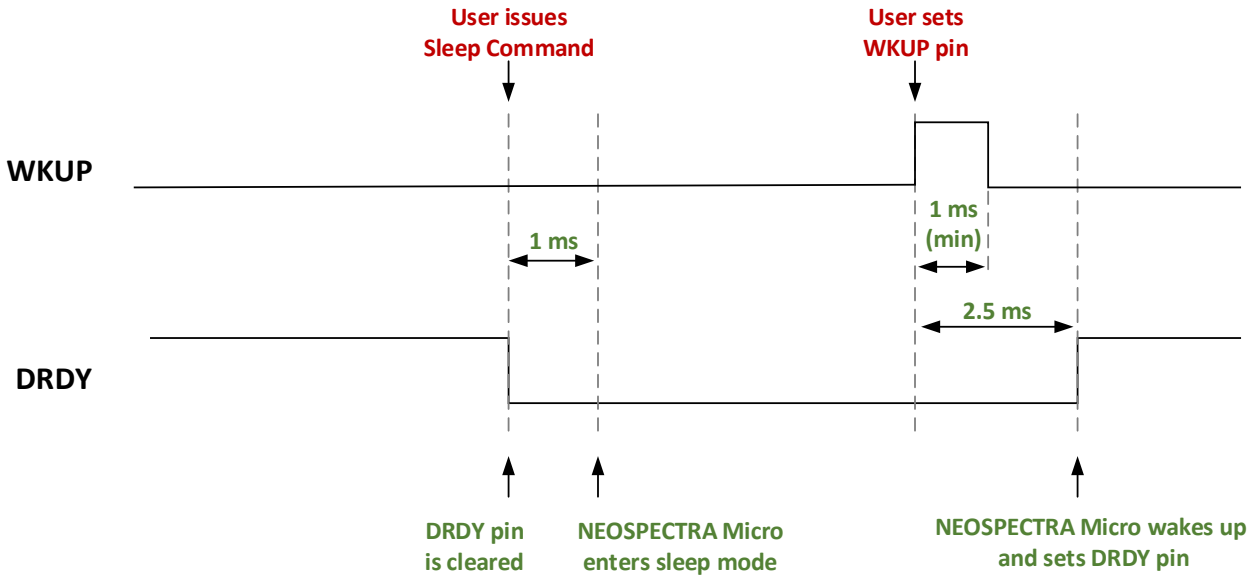


Figure 2 - Entering and Exiting Sleep Mode

3.5. Power-off Mode

NeoSpectra Micro provides an extremely low power consuming mode that can be used to power-off NeoSpectra Micro when it's not in use for extended periods of time.

Power-off mode can be entered by clearing the "EN" pin. All components are switched off in power-off mode. This provides the lowest power consumption of NeoSpectra Micro. User has to make sure that all the module interface pins are kept low as long as the module is powered-off so as to avoid any unexpected behaviors.

Power-off transition time is below 500 µs and power-off to stand-by mode switching transition time is 500 ms (maximum). The maximum current consumption of the power-off mode is 2.5 µA.

4. Electrical Characteristics

4.1. DC Characteristics

Parameter	Voltage (V)		
	Min	Typ	Max
3.3V Power Supply	3	3.3	4.5
V_{IL} Low level input voltage	0		0.4
V_{IH} High level input voltage	2.9		3.3
V_{OL} Low level output voltage			0.4
V_{OH} High level output voltage	2.9		

4.2. Power Consumption

4.2.1. Power consumption across different power modes

Mode	Current (mA)		
	Min	Typ	Max
Active Mode(1)		1250	
Stand-by Mode		70	
Sleep Mode		31	
Power-off Mode		0.001	0.0025

(1) Considering three light sources are used.

4.2.2. Sleep and Wakeup Transition Times

Mode	Mode Activation Time (ms)	Wakeup Transition Time (ms)
Sleep mode	1	2.5
Power-off mode	0.5	500

5. Communication Modes:

5.1. SPI Slave Physical Layer

NeoSpectra Micro features a simple widely-used SPI slave interface that enables handy communication with any host through the host's SPI master. NeoSpectra Micro's SPI slave has two speed modes; normal mode with frequency up to 1MHz and a high speed mode that boosts frequency up to 20MHz.

SPI_MODSEL is an output pin that indicates in which mode NeoSpectra Micro module is operating. Both modes supports the commonly-used SPI mode 0 (CPOL = 0 and CPHA = 0) and mode 3 (CPOL = 1 and CPHA = 1).

Note: A single NeoSpectra Micro sensor is either working in "Normal SPI mode" or "High Speed SPI mode". **Not both!**

The host system should accommodate both modes and select between them based on SPI_MODSEL pin after powering up.

There are no constraints on the minimum frequency to be used in either "Normal SPI mode" or "High Speed SPI mode".

Through NeoSpectra Micro's SPI slave, user can write and read an internal register file that includes all registers user may need for communication with the module: configuration registers, operation controlling registers and input/output data streaming registers.

A set of timing diagrams showing the frame structure of the SPI in the different modes accompanied with a detailed sequence of operation for each are included in the next subsections.

5.1.1. SPI Normal Mode (Up to 1MHz)

6.1.1.1 For user to write a certain register:

0- Make sure that DRDY pin (or register) is 1.

1- Open a communication frame (CSB=0).

2- The first byte is the (command + address) byte. Format it as below:

Command	Address						
0 → write	AD6	AD5	AD4	AD3	AD2	AD1	AD0

Bit 7 Bit 0

3- Transmit the required data to be written in the following bytes.

4- Close the communication frame (CSB=1).

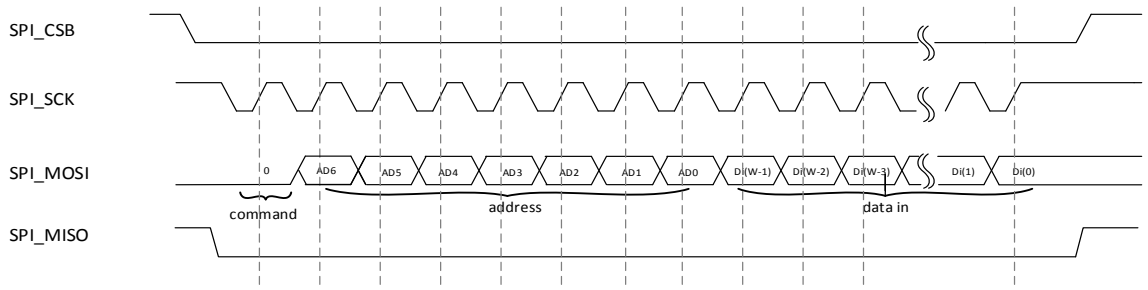


Figure 3 - SPI Write Frame Structure - Normal Mode

6.1.1.2 For user to read a certain register:

1- Open a communication frame (CSB=0).

2- The first byte is the (command + address) byte. Format it as below:

Command	Address						
1 → read	AD6	AD5	AD4	AD3	AD2	AD1	AD0

Bit 7 Bit 0

3- Transmit a number of dummy bytes equal to the number of bytes to be read in the following frame bytes + 1.

4- The data to be read will be available starting from the second byte of the dummy bytes (i.e. the third byte in total).

5- Close the communication frame (CSB=1).

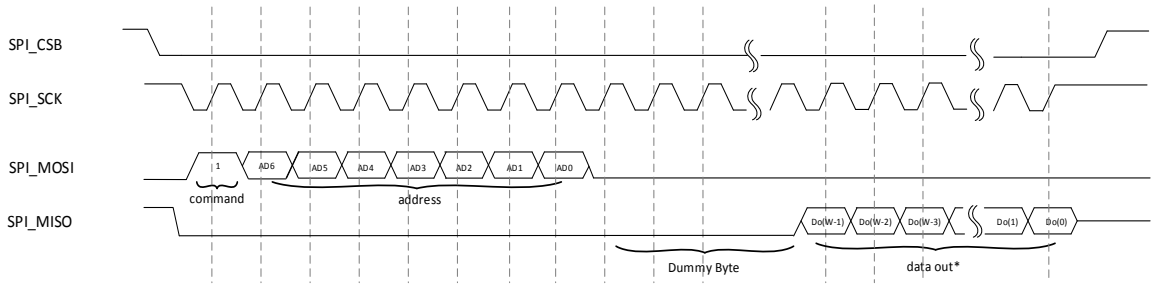


Figure 4 - SPI Read Frame Structure - Normal Mode

5.1.2. SPI High Speed Mode (Up to 20MHz – Under Development)

6.1.2.1 For user to write a certain register: (The same as the normal mode)

0- Make sure that DRDY pin (or register) is 1.

1- Open a communication frame (CSB=0).

2- The first byte is the (command + address) byte. Format it as below:

Command	Address						
0 → write	AD6	AD5	AD4	AD3	AD2	AD1	AD0

Bit 7 Bit 0

3- Transmit the required data to be written in the following bytes.

4- Close the communication frame (CSB=1).

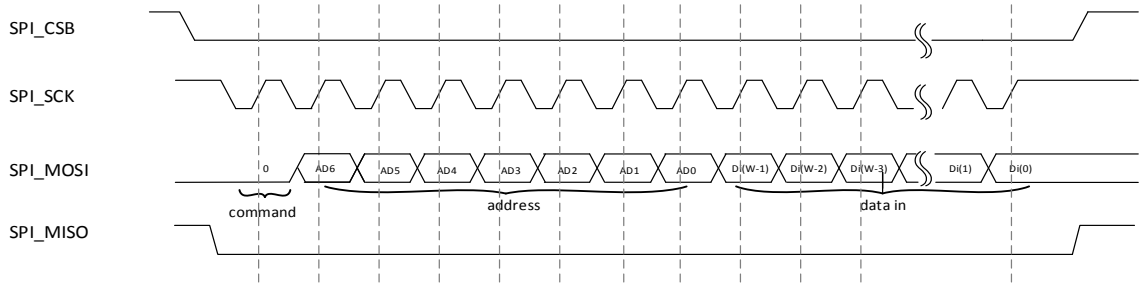


Figure 5 - SPI Write Frame Structure - High Speed Mode

6.1.2.2 For user to read a certain register:

1- Open a communication frame (CSB=0)

2- The first byte is the (command + address) byte. Format it as below:

Command	Address						
1 → read	AD6	AD5	AD4	AD3	AD2	AD1	AD0

Bit 7 Bit 0

3- Transmit a number of dummy bytes equal to the number of bytes to be read in the following frame bytes.

4- The data to be read will be available starting directly from the first byte of the dummy bytes (i.e. the second byte in total).

5- Close the communication frame (CSB=1).

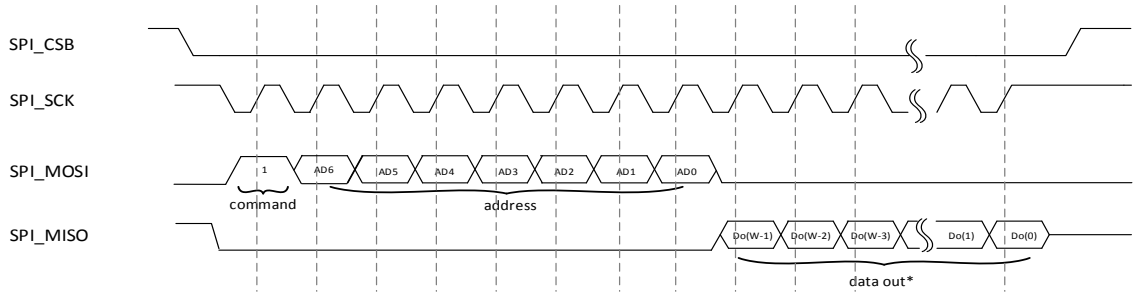


Figure 6 - SPI Read Frame Structure - High Speed Mode

5.2. Registers Description

(= Register Possible Values)

Register Name	Width (Bits)	Description	Type	Address	Offset (Bits)	Fixed-Point Quantization Length (Bits)[1]
MODULE_ID	64	Module Part number.	R	0	0	
AUTO_INCB	1	Address Auto increment enable (Active Low). 0 → Auto increment feature is enabled, which means multiple addresses of register file can be accessed in a single frame. 1 (default) → Auto increment feature is disabled, which means only one address can be accessed in a single frame.	R/W	12	0	
SNGL_CNT_MODE	4	Choose between the single and continuous modes of scanning. The continuous mode boosts the speed of scanning during specific period. (only valid during ACQUIRE_PSD & RUN_ SPECTRUM _SAMPLE commands) 0 (default) → single mode scanning. 4 → continuous mode scanning.	R/W	13	1	
XZP	2	Selects the zero padding option which is used to specify FFT number of points. 0 (default) or 1 → 1X (8k points). 2 → 2X (16k points). 3 → 4X (32k points).	R/W	13	5	
EN_COMMON_WAVE	1	Enables the Common wave number feature (linear interpolation). 0 (default) → disabled. 1 → enabled.	R/W	13	7	
UNIT_CONV	1	Selects the unit of the Wavenumber. 0 (default)→ wavenumber. 1→ wavelength.	R/W	14	0	

[1] This field indicates whether the register represent a fixed-point value or a normal value. If it's a fixed-point value , then
the corresponding double-precision number = register value / $2^{quatization\_length}$

OPT_GAIN_SET_SEL	2	Selects which optical gain settings to use during the scan. 0 (default) → Flashed optical gain settings 1 → last calculated optical gain settings 2 → External optical gain settings (the value to be used must be written in OPT_GAIN_SET_EXT register)	R/W	14	1	
WIN_SEL	3	Selects the used window for apodization. 0 (default) → No window is applied (boxcar). 1 → Gaussian. 2 → Happ-Genzel. 3 → Lorenz. 4 → External coefficients.	R/W	14	3	
ABSORBANCE	1	Enables absorbance mode (only valid in RUN_ABSORBANCE_SAMPLE operation). 0 (default) → reflectance units are enabled. 1 → absorbance units are enabled.	R/W	14	6	
SCAN_TIME	24	This register defines the scan time in milliseconds.	R/W	16	0	
PSD_NO_POINTS	13	Selects the number of points of the PSD and corresponding wave number (Up to 4K samples) from the following steps: 65, 129, 257, 513, 1024, 2048 and 4096 samples. Any other value can be written in the register, but it will be rounded to the nearest value of the previous steps. *valid only in case EN_COMMON_WAVE = 1.	R/W	20	0	
PSD_LENGTH	13	Output PSD length (in samples) (Up to 4 K samples).	R	22	0	
INITIATE_OPERATION	8	Writing this register initiates a certain operation that corresponds to the written code.	R/W	24	0	
1 → ACQUIRE_PSD		Initiates PSD acquisition operation.				
2 → RUN_SELF_CORR		Initiates self-correction routine. This routine will not write the result on Flash. The results will be stored on a volatile SRAM. It will not be kept after power down unless written on flash by PGM_SELF_CORR_COEFF operation.				
3 → RUN_REF_MTR_CORR_BG		Initiates background reading as the initial step to perform reference material correction.				

4 → RUN_REF_MTR_CORR		**Initiates reference material sample reading and run reference material correction routine.** **This routine will not write the result on Flash.** **The results will be stored on a volatile SRAM. It will not be kept after power down unless written on flash by PGM_REF_MTR_COEFF operation.** ***must be done after RUN_REF_MTR_CORR_BG command.** ***material peaks must be written in REF_MTR_WELL_0, REF_MTR_WELL_1, REF_MTR_WELL_2, REF_MTR_WELL_3 and REF_MTR_WELL_4 based on the number of required peaks.**				
5 → RUN_OPT_GAIN_ADJST		**Initiates the Optical gain adjustment routine. After its completion, the results will be stored on OPT_GAIN_SET_OUT register.** **This routine will not write the result on Flash.** **The results will be stored on a volatile SRAM. It will not be kept after power down unless written on flash by PGM_OPT_GAIN_SET operation.**				
6 → Sleep		**Requests the device to enter the "Sleep Mode".** ***Sleep Mode can be left by asserting 1 on Wake-up pin.**				
7 → WR_WIN_REQ		**External apodization coefficients write request.**				
8 → RD_PSD_WVN_REQ		**PSD or WVN read request.**				
11 → PGM_SELF_CORR_COEFF		**Stores the results of Self Correction routine on flash.**				
12 → PGM_REF_MTR_COEFF		**Stores the results of Reference Material Correction routine on flash.**				
13 → PGM_OPT_GAIN_SET		**Stores the results of Optical gain adjustment routine on flash.**				
14 → PGM_WIN_PRF		**Stores apodization window profile on flash.**				
15 → RESTORE_FACTORY_CORR		**Restore the original values of Self-Correction, Reference Material Correction and Optical Gain Adjustment. And clear any values the user previously stored on flash.**				
16 → RUN_SPECTRUM_BG		**Performs a background scan.**				
17 → RUN_ SPECTRUM _SAMPLE		**Performs a sample scan and calculate the absorbance or reflectance.** ***must be done after RUN_SPECTRUM_BG command.**				
18 → PGM_CON		**Stores the context data of the sensor so you can restore the same state of the sensor after powering off.**				

19 → RESTORE_WIN_PRF		Restores apodization window profile from flash.				
20 → RESTORE_CON		Restores context data from flash.				
21 → UPDATE_FW		Performs update for the firmware on the sensor. *must be done after transferring the firmware itself using WR_FW_REQ operation.				
22 → WR_FW_REQ		Write chunks of firmware followed by 32-bit CRC. Chunk size is 32 Kbytes (32 Kbytes - 4 bytes for fw page, and the last 4 bytes is the CRC of the fw page). Maximum number of firmware pages to be written is 8 pages.				
ABORT_OPERATION	1	This register is used to abort any ongoing operation and will put the sensor to the stand-by Mode.	WO	28	0	
SPCTRM_DATA_OUT	8	Calculated Spectrum, Absorbance or Reflectance.	R	32	0	33
FW_VERSION	32	Firmware version of the sensor.	R	36	0	
WAVE_NUM_DATA_OUT	8	Calculated Wavenumber or Wavelength.	R	40	0	30
SOURCE_LAMPS_COUNT	8	Number of lamps to operate from the source. 0 (default) → no source enable signals are active 1 → one source enable signal is active. 2 → two source enable signals are active.	R/W	41	0	
SOURCE_LAMP_SEL	8	This register determines which source enable signal to operate. 0 (default)→ source enable signal 0 is active (one lamp is ON). 1 → source enable signal 1 is active (two lamps are ON). *valid only In case SOURCE_LAMPS_COUNT = 1	R/W	42	0	
SOURCE_DELTA_T	8	Delay time between opening/closing the two source enable signals in 50 ms unit. 0 (default), 1 or 2 → 100ms. 3 → 150ms. 4 → 200ms. etc… *valid only in case SOURCE_LAMPS_COUNT = 2	R/W	43	0	
SOURCE_T1	8	Delay time after opening the source in 50 ms unit (used to adjust the settling time of the source). 0 (default) → 0 ms.	R/W	44	0	

		1 → 50 ms. **2 → 100ms.** **etc…**				
SOURCE_T2_C1	**8**	**Delay time after closing the source in 50 ms unit (used to adjust the cooling time of the source).** **0 (default) → 0 ms.** **1 → 50 ms.** **2 → 100ms.** **etc…** ***This register is only used in case that (SOURCE_T2_TMAX * 100) > SCAN_TIME**	**R/W**	**45**	**0**	
SOURCE_T2_C2	**8**	**Delay time after closing the source in percentage ratio of the scan time (used to adjust the cooling time of the source).** **0 (default) → 0% scan time in ms.** **1 → 1% scan time in ms.** **2 → 2% scan time in ms.** **etc…** ***This register is only used in case that (SOURCE_T2_TMAX * 100) <= SCAN_TIME**	**R/W**	**46**	**0**	
SOURCE_T2_TMAX	**8**	**Time boundary for deciding whether to choose SOURCE_T2_C1 or SOURCE_T2_C2 as cooling time source.** **0 (default) → 0 ms.** **1 → 100 ms.** **2 → 200 ms.** **etc…**	**R/W**	**47**	**0**	
GENERIC_DATA_OUT_LEN	**16**	**Other data out Length register (in samples).**	**R**	**48**	**0**	
GENERIC_DATA_OUT	**8**	**Other generic data output register.**	**R**	**50**	**0**	**Based on data type**
STATUS	**32**	**Status of the requested operation. To be checked after DRDY = 1 to know whether the requested operation completed successfully or not.** **0 (default)→ No error.** **Any other value → error.** ***Please refer to section 5.3 for more info**	**R**	**56**	**0**	
DRDY	**1**	**When '1' Indicates no operation is in progress. User shouldn't perform any action while register is**	**R**	**60**	**0**	

		0. 0 → Sensor is busy. User should wait. 1 (default) → Sensor is ready for actions.				
INTRPT	1	Indicates whether an error has occurred in the requested operation or not. 0 (default) → No error. 1 → error.	R	60	1	
REF_MTR_WELL_0	32	Reference material well 0 used in reference material correction routine. *(if = zero → will not be used).	R/W	64	0	20
REF_MTR_WELL_1	32	Reference material well 1 used in reference material correction routine. *(if = zero → will not be used).	R/W	68	0	20
REF_MTR_WELL_2	32	Reference material well 2 used in reference material correction routine. *(if = zero → will not be used).	R/W	72	0	20
REF_MTR_WELL_3	32	Reference material well 3 used in reference material correction routine. *(if = zero → will not be used).	R/W	76	0	20
REF_MTR_WELL_4	32	Reference material well 4 used in reference material correction routine. *(if = zero → will not be used).	R/W	80	0	20
GENERIC_DATA_IN_LEN	16	Determines the length of external stream in vector (in samples).	R/W	84	0	
GENERIC_DATA_IN	8	Incrementally accepts different stream in data after requesting its corresponding command.	R/W	86	0	Based on data type
OPT_GAIN_SET_EXT	16	This register defines the settings of the optical gain adjustment routine and is divided as follows: 3 bits: Current range settings. 3 bits: PGA1 settings. 3 bits: PGA2 settings. 7 bits: reserved.	R/W	92	0	
OPT_GAIN_SET_OUT	16	This register holds the output of the Optical gain adjustment routine (RUN_OPT_GAIN_ADJST operation).	R	94	0	

Table 2 - Registers Description

5.3. Error codes

NeoSpectra Micro error codes ranges from 1 to 127. The detailed description of these error codes is shown in the table below.

Error code	Description
0	No error
1 ~ 2	SPI communication failure
3	Flash communication failure
4 ~ 5	SPI communication failure
6 ~ 11	Reserved
12	Scan time limit error
13	Invalid sensor ID
14	Sensor not initialized
15 ~ 16	Sensor busy
17 ~ 18	Sensor configuration data is corrupt
19 ~ 27	Reserved
28	Optical settings configuration is invalid
29	Not enough memory
30 ~ 47	Sensor timeout error
48	Invalid memory address access
49	CRC check failure
50	Security check failure
51 ~ 56	Flash accessing failure
57 ~ 58	Reserved
59	SPI address not recognized
60 ~ 79	Processing error
80	Action aborted error
81 ~ 82	User interface communication failure
83 ~ 84	Watchdog timer failure
85 ~ 96	Processing error
97	Runs limit error
98	User interface communication failure
99	Reserved
100	Processing error
101	Reserved
102 ~ 105	Processing error
106 ~ 127	Reserved

5.4. Operations Guide

5.4.1. General Rules

- Writing in register file is valid only as long as DRDY = 1 except for ABORT_OPERATION
- Once a new operation is requested (INITIATE_OPEARTION register is written) DRDY goes to '0' and writing is not valid until operation is ended.
- Streaming in/out data has to be done in one single frame with AUTO_INCB = 1
- INTRPT register/pin can be used to track status for the on-going operation or after operation ends.

5.4.2. Detailed Examples for Different Operations

Below are examples of different user operations described in detailed steps.

- **RUN_OPT_GAIN_ADJST operation to apply gain adjustment routine:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers
   - SCAN_TIME = time in milliseconds (e.g. 2000).
   - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
   - SOURCE_T1 = 14 (for 700 ms settling time).
   - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
   - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
   - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
   - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_OPT_GAIN_ADJST" code (5) in INITIATE_OPERATION register
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
5- Read STATUS register to check the status of the operation
6- Write OPT_GAIN_SET_SEL = 1 to use the calculated gain value in the upcoming measurements.

- **PGM_OPT_GAIN_SET operation to store the result of gain adjustment routine on flash:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write "PGM_OPT_GAIN_SET" code (13) in INITIATE_OPERATION register
3- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4- Read STATUS register to check the status of the operation
5- Write OPT_GAIN_SET_SEL = 0 to use the flashed gain value in the upcoming measurements.

- **ACQUIRE_PSD operation for getting the power spectral density:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers (for example)
    - SCAN_TIME = time in milliseconds (e.g. 2000).
    - XZP = zero padding option (e.g. 0 → 8K points).
    - WIN_SEL = apodization window option (e.g. 0 → boxcar).
    - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
    - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
    - SOURCE_T1 = 14 (for 700 ms settling time).
    - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
    - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
    - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
    - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "ACQUIRE_PSD" code (1) in INITIATE_OPERATION register.
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it).
5- Read STATUS register to check the status of the operation.
6- Read PSD_LENGTH register.
7- Write AUTO_INCB = 1.
8- Read the PSD from SPCTRM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
9- Close the communication frame
10- Read the Wave number vector from WAVE_NUM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
11- Close the communication frame

- **ACQUIRE_PSD operation for getting the power spectral density with continuous mode:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers (for example)
    - SCAN_TIME = time in milliseconds (e.g. 2000).
    - SNGL_CNT_MODE = 4 (for selecting continuous scanning mode).
    - XZP = zero padding option (e.g. 0 → 8K points).
    - WIN_SEL = apodization window option (e.g. 0 → boxcar).
    - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
    - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
    - SOURCE_T1 = 14 (for 700 ms settling time).
    - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
    - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
    - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
    - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "ACQUIRE_PSD" code (1) in INITIATE_OPERATION register.
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it).
5- Read STATUS register to check the status of the operation.

6- Read PSD_LENGTH register.
7- Write AUTO_INCB = 1.
8- Read the PSD from SPCTRM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
9- Close the communication frame
10- Read the Wave number vector from WAVE_NUM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
11- Close the communication frame
12- After reading both PSD & WVN, DRDY register (and pin) should automatically go to '0'. So the user should loop on the steps from 4→12 to acquire further scans without the need for initiating new commands.
13- For **exiting** the continuous mode, write SNGL_CNT_MODE = '0'. Then acquire one last PSD & WVN. DRDY register (and pin) should stay '1' after that last acquisition until any further command is requested.

- **RUN_SPECTRUM_BG operation for scanning background:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers (for example)
   - SCAN_TIME = time in milliseconds (e.g. 2000).
   - XZP = zero padding option (e.g. 0 → 8K points).
   - WIN_SEL = apodization window option (e.g. 0 → boxcar).
   - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
   - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
   - SOURCE_T1 = 14 (for 700 ms settling time).
   - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
   - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
   - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
   - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_SPECTRUM_BG" code (16) in INITIATE_OPERATION register.
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it).
5- Read STATUS register to check the status of the operation.

- **RUN_SPECTRUM_SAMPLE operation for getting the spectrum of a testing material (RUN_SPECTRUM_BG should be executed at least once before this operation):**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers (for example)
   - SCAN_TIME = time in milliseconds (e.g. 2000). ***It's recommended to choose the same value of the background.***
   - XZP = zero padding option (e.g. 0 → 8K points). ***Must be the same value of background.***
   - WIN_SEL = apodization window option (e.g. 0 → boxcar). ***It's recommended to choose the same value of the background.***
   - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
   - ABSORBANCE = 1 (to get spectrum in absorbance units)
   - SOURCE_LAMPS_COUNT = 2 (for full light source enable).

- SOURCE_T1 = 14 (for 700 ms settling time).
- SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
- SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
- SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
- SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).

3- Write "RUN_ SPECTRUM _SAMPLE" code (17) in INITIATE_OPERATION register
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
5- Read STATUS register to check the status of the operation
6- Read PSD_LENGTH register
7- Write AUTO_INCB = 1.
8- Read the spectrum from SPCTRM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
9- Close the communication frame
10- Read the Wave number vector from WAVE_NUM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
11- Close the communication frame

- **RUN_SPECTRUM_SAMPLE operation for getting the spectrum of a testing material (RUN_SPECTRUM_BG should be executed at least once before this operation) with continuous mode:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers (for example)
   - SCAN_TIME = time in milliseconds (e.g. 2000). ***It's recommended to choose the same value of the background.***
   - SNGL_CNT_MODE = 4 (for selecting continuous scanning mode).
   - XZP = zero padding option (e.g. 0 → 8K points). ***Must be the same value of background.***
   - WIN_SEL = apodization window option (e.g. 0 → boxcar). ***It's recommended to choose the same value of the background.***
   - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
   - ABSORBANCE = 1 (to get spectrum in absorbance units)
   - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
   - SOURCE_T1 = 14 (for 700 ms settling time).
   - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
   - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
   - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
   - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_ SPECTRUM _SAMPLE" code (17) in INITIATE_OPERATION register
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
5- Read STATUS register to check the status of the operation
6- Read PSD_LENGTH register
7- Write AUTO_INCB = 1.

8- Read the spectrum from SPCTRM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
9- Close the communication frame
10- Read the Wave number vector from WAVE_NUM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
11- Close the communication frame
12- After reading both PSD & WVN, DRDY register (and pin) should automatically go to '0'. So the user should loop on the steps from 4→12 to acquire further scans without the need for initiating new commands.
13- For **exiting** the continuous mode, write SNGL_CNT_MODE = '0'. Then acquire one last PSD & WVN. DRDY register (and pin) should stay '1' after that last acquisition until any further command is requested.

- **RD_PSD_WVN_REQ This operation enables you request PSD, wavenumber or spectrum for reading:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write "RD_PSD_WVN_REQ" code (8) in INITIATE_OPERATION register
3- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4- Read STATUS register to check the status of the operation
5- Read PSD_LENGTH register
6- Write AUTO_INCB = 1.
7- Read the spectrum from SPCTRM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
8- Close the communication frame
9- Read the Wave number vector from WAVE_NUM_DATA_OUT register successively in one frame with the number of samples determined by PSD_LENGTH
10- Close the communication frame

Note: RD_PSD_WVN_REQ operation is not required if you read SPCTRM_DATA_OUT and WAVE_NUM_DATA_OUT registers just after ACQUIRE_PSD or RUN_SPECTRUM_SAMPLE operations.

- **RUN_SELF_CORR operation to apply self-correction routine:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers
    - SCAN_TIME = time in milliseconds (e.g. 2000).
    - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
    - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
    - SOURCE_T1 = 14 (for 700 ms settling time).
    - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
    - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
    - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
    - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_SELF_CORR" code (2) in INITIATE_OPERATION register
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)

5- Read STATUS register to check the status of the operation

- **PGM_SELF_CORR_COEFF operation to store the result of self-correction routine on flash:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin
2- Write "PGM_SELF_CORR_COEFF" code (11) in INITIATE_OPERATION register
3- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4- Read STATUS register to check the status of the operation

- **RUN_REF_MTR_CORR_BG operation to apply reference material correction routine (background scan):**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers
    - SCAN_TIME = time in milliseconds (e.g. 2000).
    - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
    - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
    - SOURCE_T1 = 14 (for 700 ms settling time).
    - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
    - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
    - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
    - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_REF_MTR_CORR_BG" code (3) in INITIATE_OPERATION register
4- Poll on DRDY register till it becomes '1' or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
5- Read STATUS register to check the status of the operation

- **RUN_REF_MTR_CORR operation to apply reference material correction routine (reference material scan):**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY pin.
2- Write the needed configuration registers
    - SCAN_TIME = time in milliseconds (e.g. 2000).
    - OPT_GAIN_SET_SEL = 1 (to make use of the last calculated gain value).
    - REF_MTR_WELL_0 = reference material peak0 wavelength.
    - REF_MTR_WELL_1 = reference material peak1 wavelength (if any, 0 otherwise).
    - REF_MTR_WELL_2 = reference material peak2 wavelength (if any, 0 otherwise).
    - REF_MTR_WELL_3 = reference material peak3 wavelength (if any, 0 otherwise).
    - REF_MTR_WELL_4 = reference material peak4 wavelength (if any, 0 otherwise).
    - SOURCE_LAMPS_COUNT = 2 (for full light source enable).
    - SOURCE_T1 = 14 (for 700 ms settling time).
    - SOURCE_DELTA_T = 2 (for 100 ms delay between the two sources).
    - SOURCE_T2_TMAX = 10 (for 1000 ms time boundary of cooling time source).
    - SOURCE_T2_C1 = 5 (for 250 ms cooling time, in case scan time < 1000 ms).
    - SOURCE_T2_C2 = 35 (for 35% of scan time as cooling time, in case scan time >= 1000 ms).
3- Write "RUN_REF_MTR_CORR" code (4) in INITIATE_OPERATION register

4- Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
5- Read STATUS register to check the status of the operation

- **PGM_REF_MTR_COEFF operation to store the result of reference material correction routine on flash:**

1- Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2- Write “PGM_REF_MTR_COEFF” code (12) in INITIATE_OPERATION register
3- Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4- Read STATUS register to check the status of the operation

- **WR_WIN_REQ operation to write the polynomial coefficients of an apodization window:**

1- Poll on DRDY register until it becomes ‘1’ or wait for the DRDY interrupt pin
2- Write the data stream length in samples in GENERIC_DATA_IN_LEN register
3- Write “WR_WIN_REQ” code (7) in initiate operation register
4- Poll on DRDY register till it becomes '1' or use DRDY interrupt pin
5- Read STATUS register to check the status of the operation if INTRPT = 1
6- Write AUTO_INCB = 1
7- Write the window coefficients through GENERIC_DATA_IN register successively in one frame
8- Close the communication frame
9- Poll on DRDY reg/pin = 1 indicating entered data has been accepted
10- Read STATUS register to check the status of the operation if INTRPT = 1

- **RESTORE_FACTORY_CORR operation to restore correction and optical gain settings to the factory settings:**

1- Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2- Write “RESTORE_FACTORY_CORR” code (15) in INITIATE_OPERATION register
3- Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4- Read STATUS register to check the status of the operation

- **PGM_WIN_PRF operation to store the apodization window profile on flash:**

1. Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2. Write “PGM_WIN_PRF” code (14) in INITIATE_OPERATION register
3. Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4. Read STATUS register to check the status of the operation

- **PGM_CON operation to store the context data on flash:**

1. Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2. Write “PGM_CON” code (18) in INITIATE_OPERATION register
3. Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4. Read STATUS register to check the status of the operation

- **RESTORE_WIN_PRF operation to restore the apodization window profile from flash:**

1. Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2. Write “RESTORE_WIN_PRF” code (19) in INITIATE_OPERATION register
3. Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4. Read STATUS register to check the status of the operation

- **RESTORE_CON operation to restore the context data from flash:**

1. Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2. Write “RESTORE_CON” code (20) in INITIATE_OPERATION register
3. Poll on DRDY register till it becomes ‘1’ or wait for the DRDY pin (If INTRPT reg/pin is set during this waiting period, this indicates a warning, read STATUS register to check it)
4. Read STATUS register to check the status of the operation

- **Sleep operation to put the sensor in the sleep mode:**

1- Poll on DRDY register until it becomes ‘1’ or wait for the DRDY pin
2- Write “Sleep” code (6) in INITIATE_OPERATION register

- **Abort an ongoing operation:**

For user to abort an ongoing operation

1- Write “ABORT_OPERATION” register = ‘1’
2- Poll on DRDY register = ‘1’ or wait for the DRDY interrupt pin

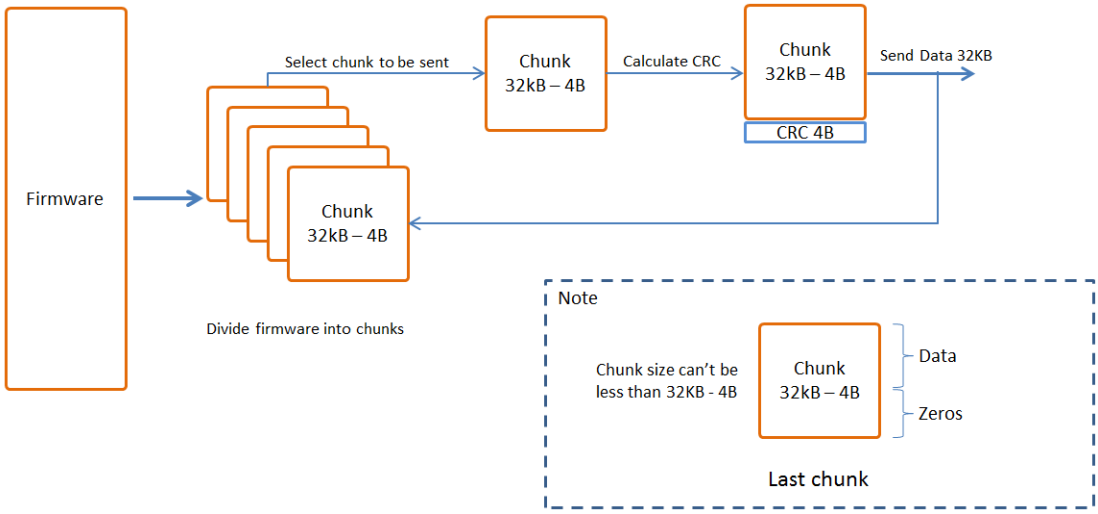
- **WR_FW_REQ operation to write a chunk of the sensor firmware:**

1- Poll on DRDY register until it becomes ‘1’ or wait for the DRDY interrupt pin
2- Write the data stream length in samples in GENERIC_DATA_IN_LEN register (e.g. for 32Kbytes chucks, write (32768 / 4))
3- Write “WR_FW_REQ” code (22) in initiate operation register
4- Poll on DRDY register till it becomes '1' or use DRDY interrupt pin
5- Read STATUS register to check the status of the operation if INTRPT = 1
6- Write AUTO_INCB = 1
7- Write the firmware through GENERIC_DATA_IN register successively in one frame
8- Close the communication frame
9- Poll on DRDY reg/pin = 1 indicating entered data has been accepted
10- Read STATUS register to check the status of the operation if INTRPT = 1

Note:

- The sensor firmware should be divided into chunks each of size 32Kbytes.
- WR_FW_REQ is used to write single chunk at a time.
- The firmware chunks must be written into order, otherwise the following UPDATE_FW operation will fail.

WR_FW_REQ flowchart



- **UPDATE_FW operation to write a chunk of the sensor firmware:**

1- Poll on DRDY register until it becomes '1' or wait for the DRDY interrupt pin
2- Write the 32-bit CRC calculate from the whole firmware into REF_MTR_WELL_0 register
3- Write "UPDATE_FW" code (21) in initiate operation register
4- Poll on DRDY interrupt pin to check whether the operation completed or still in progress **(you can't use DRDY register in this case!)**
5- Reset the sensor to launch the new firmware

Note: The 32-bit CRC used with the operations WR_FW_REQ and UPDATE_FW has the polynomial $(04C11DB7)_{16}$.

UPDATE_FW flowchart