

## nano spectrometer module for IoT

NSP32m is an ultra compact spectral sensing platform based on revolutionary digital-nano convergence technology. With built-in advanced spectral signal processing and standard SPI or UART interface, it is easy to use and can be integrated into any IoT device.

NSP32m is individually calibrated nano-accurately and provides device-to-device repeatability. Also the measured spectrum data is compatible with the spectrum data measured by conventional optical spectrum analyzers, so that the spectral databases for applications are portable and reusable.

### Features

- Ultra compact size
- Input optics built-in
- Wide field-of-view(FOV)
- Wide wavelength range
- No input fiber, no moving parts
- Individually calibrated
- SPI and UART dual interfaces
- Message-based Communication
- Low power consumption
- 3.3V single power
- Wide operating temperature
- Repeatability, device-to-device

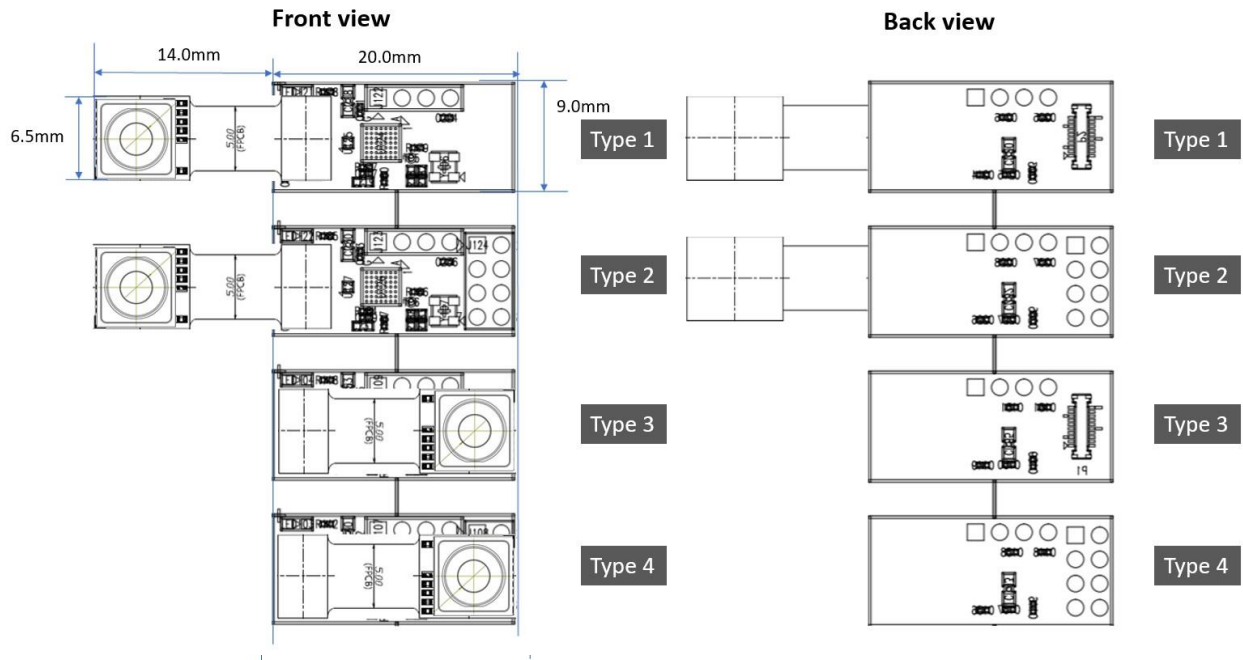
### Potential Applications

- Health, mobile/wearable
- Lab-on-the-phone, point-of-care
- Food quality, food process
- Precision agriculture
- Water quality
- Air quality, air pollution
- LED lighting
- Color, camera/display
- Cosmetics
- Security

### Key Parameters

Parameter	Value
Wavelength range	350nm ~1050nm (max.) Different types for different range
Resolution	10nm~30nm
Repeatability(peak)	1nm
SNR	> 100:1
Dynamic range	> 1000:1
Integration time	211us~, manual or auto setting
Measurement speed	55ms (min.) per spectrum measurement
Linearity	99% for optimal input power range
FOV(field of view)	> +/- 60 degree
Low power consumption	11uA @standby mode
Operation temperature	0°C ~ 60°C

## 1. Module Types and Dimension

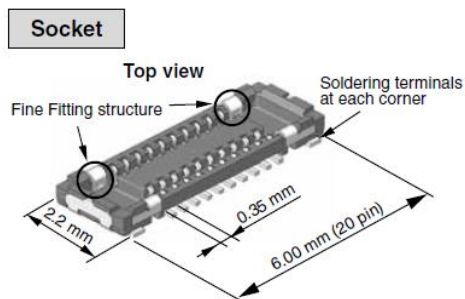


Connector Types:

Module Type	Sensor Head Direction	Connection Type	Mating Part Number
1	Outward (Flexible)	AXG820044 (20 pins)	AXG720047
2	Outward (Flexible)	2.0mm pitch holes (8 pins)	Wire or Header
3	Inward	AXG820044 (20 pins)	AXG720047
4	Inward	2.0mm pitch holes (8 pins)	Wire or Header

NOTE ) For 20pin connector, AXG720047 (Socket type) needs to be used on customer side.

[http://www3.panasonic.biz/ac/e\\_download/control/connector/base-fpc/catalog/con\\_eng\\_a35us.pdf](http://www3.panasonic.biz/ac/e_download/control/connector/base-fpc/catalog/con_eng_a35us.pdf)



## 2. Pin Assignment and Transmission Modes

NSP32m uses the same port for SPI and UART connection.

### 2.1 SPI Mode

– AXG820044 (20 Pins connector)

Pin	Signal	I/O	Pin	Signal	I/O
F1	GND	I	F2	GND	I
1	NC	reserved	2	VDD3V3	I
3	SPI_SCK	I	4	VDD3V3	I
5	SPI_MISO	O	6	VDD3V3	I
7	SPI_MOSI	I	8	GND	I
9	SPI_NSS	I	10	GND	I
11	Wakeup/Reset	I	12	GND	I
13	Ready trigger	O	14	NC	reserved
15	VDD3V3	I	16	NC	reserved
17	VDD3V3	I	18	NC	reserved
19	VDD3V3	I	20	NC	reserved
F3	GND	I	F4	GND	I

\*NC: No Connection

– Through Hole type (2.0mm pitch, 8 pins in 2 rows wire or header)

Pin	Signal	I/O	Pin	Signal	I/O
1	VDD3V3	I	2	SPI_SCK	I
3	GND	I	4	SPI_MISO	O
5	Ready trigger	O	6	SPI_MOSI	I
7	Wakeup/Reset	I	8	SPI_NSS	I

- 1) Wakeup/Reset: Input with pull-up. Pull low for min. 50us to do Wakeup/Reset. Please refer to section 4.2 for timing diagram.
- 2) Ready trigger: Output with push-pull. Falling edge to 1) indicate the reboot process is completed and NSP32m is ready to use, or 2) indicate the measurement data is ready to be retrieved. Please see commands 'CMD\_ACQ\_SPECTRUM', 'CMD\_GET\_SPECTRUM', 'CMD\_ACQ\_XYZ', and "CMD\_GET\_XYZ" in section 3 and refer to section 4.7 and 4.8 for timing diagram.
- 3) All I/O pins except VDD and GND are at high impedance when NSP32m is in Standby mode.

Note ) A level-shift circuit might be required If connecting to a 'non 3V3 logic level' MCU/device such as Arduino.

## 2.2 UART Mode

– AXG820044 (20 Pins connector)

Pin	Signal	I/O	Pin	Signal	I/O
F1	GND	I	F2	GND	I
1	NC	reserved	2	VDD3V3	I
3	UART_RX	I	4	VDD3V3	I
5	BAUD_SELECTION[1]	I	6	VDD3V3	I
7	BAUD_SELECTION[0]	I	8	GND	I
9	UART_TX	O	10	GND	I
11	Wakeup/Reset	I	12	GND	I
13	Ready trigger	O	14	NC	reserved
15	VDD3V3	I	16	NC	reserved
17	VDD3V3	I	18	NC	reserved
19	VDD3V3	I	20	NC	reserved
F3	GND	I	F4	GND	I

– Through Hole type (2.0mm pitch, 8 pins in 2 rows wire or header)

Pin	Signal	I/O	Pin	Signal	I/O
1	VDD3V3	I	2	UART_RX	I
3	GND	I	4	BAUD_SELECTION[1]	I
5	Ready trigger	O	6	BAUD_SELECTION[0]	I
7	Wakeup/Reset	I	8	UART_TX	O

- 1) To select baud rate, set the BAUD\_SELECTION[0], BAUD\_SELECTION[1] at 0 or 1 (see table below). The selected baud rate will be set at power-on or at triggering Wakeup/Reset.

BAUD_SELECTION[1]	BAUD_SELECTION[0]	Baud Rate (bits/s)
NC	NC	115200
1 (VDD)	1 (VDD)	115200
1 (VDD)	0 (GND)	38400
0 (GND)	1 (VDD)	19200
0 (GND)	0 (GND)	9600

- 2) Wakeup/Reset: Input with pull-up. Pull low for min. 50us to do Wakeup/Reset. Please refer to section 4.2 for timing diagram.
- 3) Ready trigger: Output with push-pull. Falling edge to 1) indicate the reboot process is completed and NSP32m is ready to use, or 2) indicate the measured data is ready to be retrieved. Please see commands 'CMD\_ACQ\_SPECTRUM', 'CMD\_GET\_SPECTRUM', 'CMD\_ACQ\_XYZ', and "CMD\_GET\_XYZ" in section 3 and refer to section 4.7 and 4.8 for timing diagram.
- 4) All I/O pins except VDD and GND are at high impedance when NSP32m is in Standby mode.

Note ) A level-shift circuit might be required If connecting to a ‘non 3V3 logic level’ MCU/device such as Arduino.

## 2.3 SPI Mode and UART Mode Selection

NSP32m uses the same port for SPI and UART connection, and auto detects between SPI mode and UART mode using the rule below:

SPI_SCK/UART_RX	NSP32m Mode
NC	UART
1 (HIGH)	UART
0 (LOW)	SPI

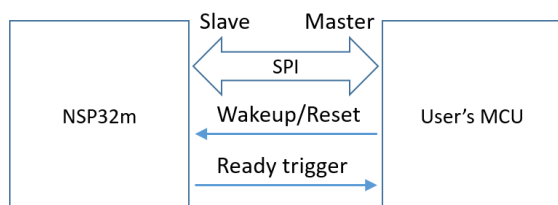
Detection period: about 25ms from upon receiving the Reset/Wakeup (or power-on) to the generating of Ready trigger. SPI\_SCK/UART\_RX needs to stay in the same status during this 25ms detection period. Please refer to section 4.1 and 4.2 for timing diagrams.

Note 1) In a typical condition, NSP32m can auto set to SPI mode when connected to SPI interface, and can auto set to UART mode when connected to UART interface, based on the mechanism that Master SPI(mode 0)@idle the SPI\_SCK is at low level whereas UART@idle the UART\_TX (UART\_RX of NSP32m) is at high level.

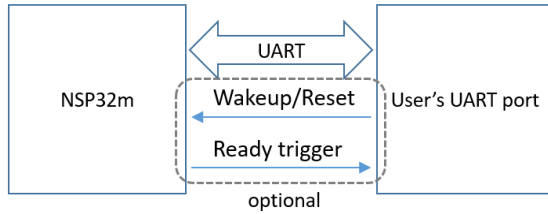
Note 2) In case NSP32m is not in the required transmission mode, connect SPI\_SCK/UART\_RX to 0 or 1 at need, and explicitly do the Wakeup/Reset operation.

## 2.4 NSP32m Connection Configuration

SPI connection (recommended)



## UART connection



- 1) SPI connection is recommended if SPI interface and GPIO(s) of connecting Ready trigger and Wakeup/Reset are available.
- 2) UART connection can be used when SPI interface or GPIO of connecting Ready trigger or Wakeup/Reset is not available, for example, for PC UART users.
- 3) NSP32m has built-in power-on reset. Please note that NSP32m shall not enter Standby mode if Wakeup/Reset is not used.
- 4) For operation without using Ready trigger, please refer to section 3.4.6 for more information.

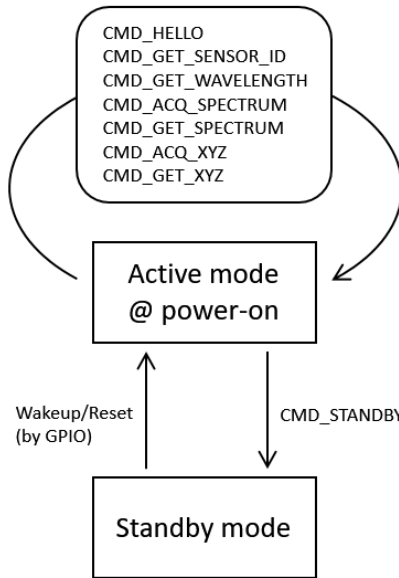
## 3. NSP32m Commands and Operations

### 3.1 NSP32m Command overview

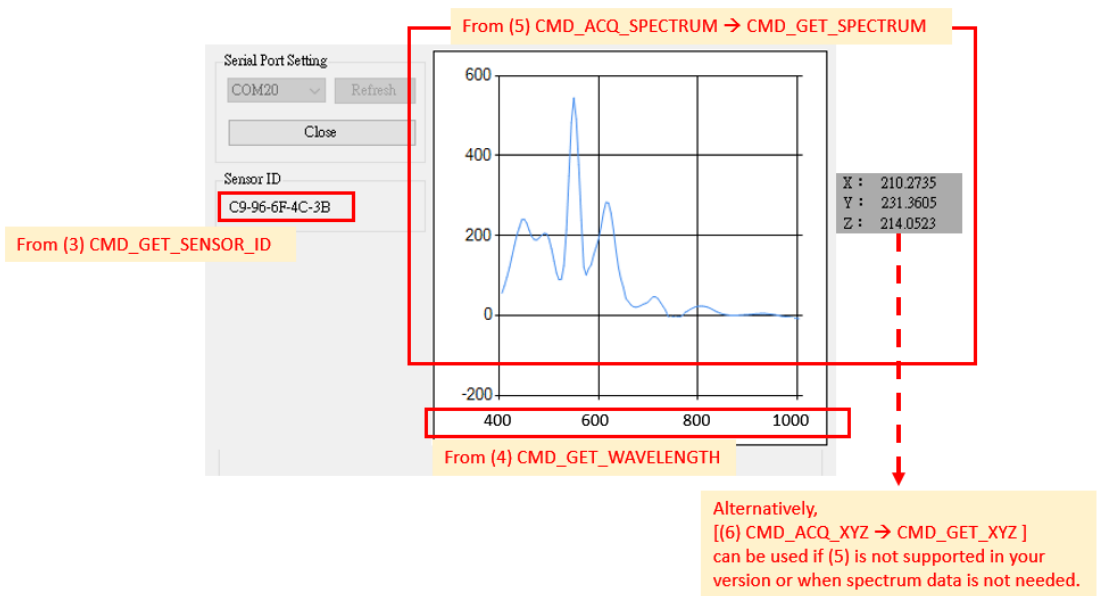
Command Name	Function code	Description	Command Packet Length (bytes)	Return Packet Length (bytes)
CMD_HELLO	0x01	Used for the SPI link check or UART link check	5	5
CMD_STANDBY	0x04	Used for NSP32m to enter Standby mode	5	5
CMD_GET_SENSOR_ID	0x06	Used for retrieving the sensor's ID.	5	10
CMD_GET_WAVELENGTH	0x24	Used for retrieving the wavelength positions (in the unit of 'nm')	5	279
CMD_ACQ_SPECTRUM	0x26	Used for initiating the spectrum measurement	10	5
CMD_GET_SPECTRUM	0x28	Used for retrieving the measured spectrum data	5	565
CMD_ACQ_XYZ	0x2A	Used for initiating the CIE 1931 XYZ measurement	10	5
CMD_GET_XYZ	0x2C	Used for retrieving the measured CIE 1931 XYZ values.	5	21

## 3.2 NSP32m Operation Modes:

Status diagram between Standby mode and Active mode



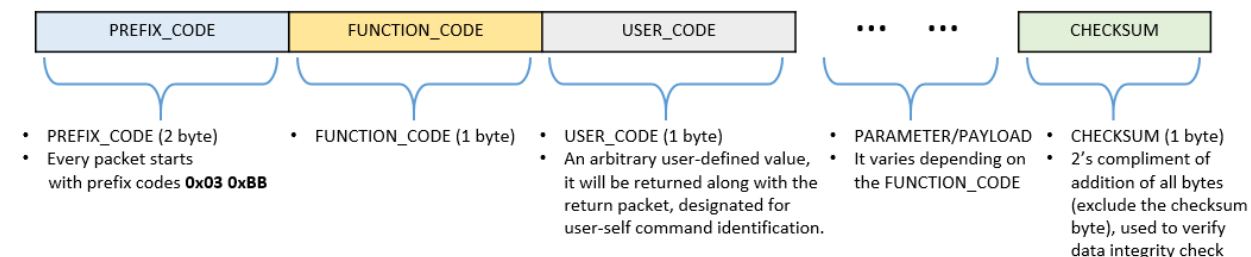
## 3.3 NSP32m Typical Operation Sequence and Example



Note ) Please check [www.nanolambda.com](http://www.nanolambda.com) for example source codes and more information. The above GUI example is based on the PC→UART→NSP32m example.

## 3.4 NSP32m Command Format and Descriptions

### 3.4.1 Packet Structure Overview



### C++ example code for calculating checksum

```

/**
 * calculate checksum and append it to the end of the buffer (use "modular sum" method)
 * @param pBuf buffer
 * @param len data length (excluding the checksum)
 */
void NSP32::PlaceChecksum(uint8_t *pBuf, uint32_t len)
{
    uint8_t checksum = 0;

    // sum all bytes (excluding the checksum)
    while(len-- > 0)
    {
        checksum += *pBuf++;
    }

    // take two's complement, and append the checksum to the end
    *pBuf = (~checksum) + 1;
}

```

### 3.4.2 Command 'CMD\_HELLO' (FUNCTION CODE : 0x01)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x01	0x00	0x41

- Return packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x01	0x00	0x41

Fixed value  
 Variable value

- Command 'CMD\_HELLO' is used for SPI link check or UART link check.



### 3.4.3 Command 'CMD\_STANDBY' (FUNCTION\_CODE : 0x04)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x04	0x00	0x3E

- Return packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x04	0x00	0x3E

Fixed value  
 Variable value

- Command 'CMD\_STANDBY' can be used for NSP32m entering standby mode. In SPI mode, NSP32m will enter standby mode right after the user's MCU retrieves the return packet. In UART mode, NSP32m will enter standby mode right after NSP32m send out the return package.

### 3.4.4 Command 'CMD\_GET\_SENSOR\_ID' (FUNCTION\_CODE : 0x06)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x06	0x00	0x3C

- Return packet structure example (length: 10 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	Sensor id					CHECKSUM
0x03	0xBB	0x06	0x00	0xC9	0x96	0x74	0x40	0x9B	0x8E

Fixed value  
 Variable value

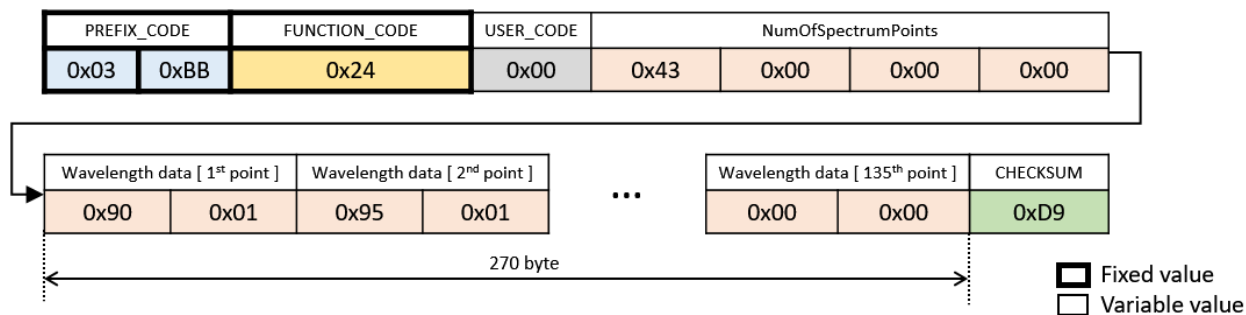
- Command 'CMD\_GET\_SENSOR\_ID' is used for retrieving the sensor ID. Every sensor has its own unique ID (5 byte), presented in the [Sensor id] bytes as "C9-96-74-40-9B" in this example.

### 3.4.5 Command 'CMD\_GET\_WAVELENGTH' (FUNCTION\_CODE : 0x24)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x24	0x00	0x1E

- Return packet structure example (length: 279 byte)



- Command 'CMD\_GET\_WAVELENGTH' is used for retrieving the wavelength positions (in the unit of 'nm'), in which:
  - [ NumOfSpectrumPoints ] : (32 bit unsigned integer, little endian, max = 135)  
Number of valid spectrum points.
  - [ Wavelength data [ i<sup>th</sup> point ] ] : (16 bit unsigned integer, little endian)  
The wavelength in the unit of 'nm' corresponding to the measured i<sup>th</sup> spectrum value.  
Note ) [ Wavelength data [ i<sup>th</sup> point ] ] = 0 when i > NumOfSpectrumPoints.

### 3.4.6 Command 'CMD\_ACQ\_SPECTRUM' (FUNCTION CODE : 0x26)

- Command packet structure example (length: 10 byte)

PREFIX_CODE		FUNCTION_CODE		USER_CODE	IntegrationTime		FrameAvgNum	EnableAE	ActiveReturn	CHECKSUM
0x03	0xBB	0x26		0x00	0x20	0x00	0xA0	0x00	0x00	0x5C

- Return packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE		USER_CODE	CHECKSUM
0x03	0xBB	0x26		0x00	0x1C

Fixed value  
 Variable value

- Command 'CMD\_ACQ\_SPECTRUM' is used for initiating the spectrum measurement, in which:
  - [ IntegrationTime ] : (16 bit unsigned integer, little endian, max = 1200)  
The length of exposure time. Exposure time (ms) = (896\*[IntegrationTime] + 160) / 500. To get a stable measurement at low light condition, IntergrationTime should be enlarged. IntergrationTime needs to be reduced if 'SaturationFlag = 1' (see 'SaturationFlag' in section 3.4.7),  
For typical usage, AE (auto-exposure integration time) is recommended.
  - [ FrameAvgNum ] : (8bit unsigned integer)  
The number of spectrum frames to be averaged during one measurement. Larger value of FrameAvgNum can deliver a more stable measurement result, but consumes more time due to more frame acquisition.
  - [ EnableAE ] : (8bit unsigned integer, 0 or 1)  
Enable or disable the AE (auto-exposure integration time). Set to 1 to enable the auto-exposure integration time for typical usage. Set to 0 to disable.

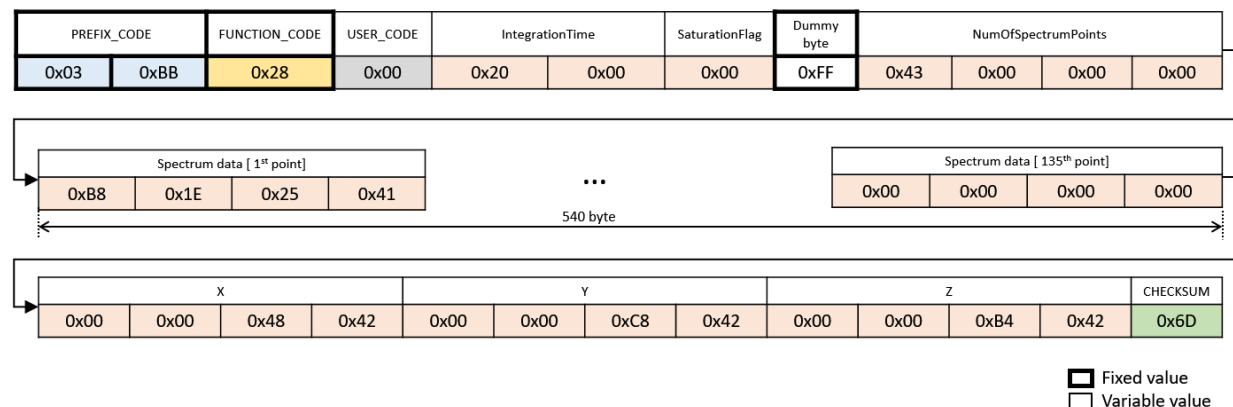
- [ ActiveReturn ] : (8bit unsigned integer, 0 or 1)  
 The flag is designed for UART mode in the situation that 'ready trigger' pin is not accessible on user side (e.g. when connecting from PC's UART). When the flag is set to 1, NSP32m actively sends out the 'CMD\_GET\_SPECTRUM' return packet once the acquisition is done. In other words, user only sends one 'CMD\_ACQ\_SPECTRUM' command packet, but gets two consecutive return packets (immediately one return packet of 'CMD\_ACQ\_SPECTRUM' with function code 0x26, and later the other return packet of 'CMD\_GET\_SPECTRUM' with function code 0x28).  
 Note ) the flag value is only effective in UART mode, and will be ignored in SPI mode.

### 3.4.7 Command 'CMD\_GET\_SPECTRUM' (FUNCTION CODE : 0x28)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x28	0x00	0x1A

- Return packet structure example (length: 565 byte)



- Command 'CMD\_GET\_SPECTRUM' is used for retrieving the measured spectrum data, initiated by 'CMD\_ACQ\_SPECTRUM', in which:
  - [ IntegrationTime ] : (16 bit unsigned integer, little endian, max = 1200)  
 A return of the used IntegrationTime. When AE is enabled, the searched intergrationTime will be returned.
  - [ SaturationFlag ] : (8bit unsigned integer, 0 or 1)  
 Indicate whether the measured spectrum is saturated or not (1 = saturated; 0 = unsaturated). If SaturationFlag = 1, the measured spectrum can be seriously distorted and should be discarded
  - [ NumOfSpectrumPoints ] : (32 bit unsigned integer, little endian, max = 135)  
 Number of valid spectrum points.
  - [ Spectrum data [ i<sup>th</sup> point ] ] : (32 bit float, little endian)  
 The measured spectrum value corresponding to the i<sup>th</sup> wavelength (nm). Also see 'CMD\_GET\_WAVELENGTH'.  
 Note ) [Spectrum data [ i<sup>th</sup> point ] ] = 0 when i > NumOfSpectrumPoints.
  - [ X ], [ Y ], [ Z ] : (32 bit float, little endian)  
 The measured tristimulus X, Y, and Z value of the CIE 1931 XYZ color space.

### 3.4.8 Command 'CMD\_ACQ\_XYZ' (FUNCTION CODE : 0x2A)

- Command packet structure example (length: 10 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	IntegrationTime		FrameAvgNum	EnableAE	ActiveReturn	CHECKSUM
0x03	0xBB	0x2A	0x00	0x20	0x00	0xA0	0x00	0x00	0x58

- Return packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x2A	0x00	0x18

Fixed value  
 Variable value

- Command 'CMD\_ACQ\_XYZ' is used for initiating the measurement for the tristimulus X, Y, and Z value of the CIE 1931 XYZ color space. Please refer to section 3.4.6 for the description of [ IntegrationTime ], [ FrameAvgNum ], [ EnableAE ], and [ ActiveReturn ]. 'CMD\_ACQ\_XYZ' is used if 'CMD\_ACQ\_SPECTRUM' is not supported in your version or when spectrum data is not needed.

### 3.4.9 Command 'CMD\_GET\_XYZ' (FUNCTION CODE : 0x2C)

- Command packet structure example (length: 5 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	CHECKSUM
0x03	0xBB	0x2C	0x00	0x16

- Return packet structure example (length: 21 byte)

PREFIX_CODE		FUNCTION_CODE	USER_CODE	IntegrationTime		SaturationFlag	Dummy byte
0x03	0xBB	0x2C	0x00	0x20	0x00	0x00	0xFF

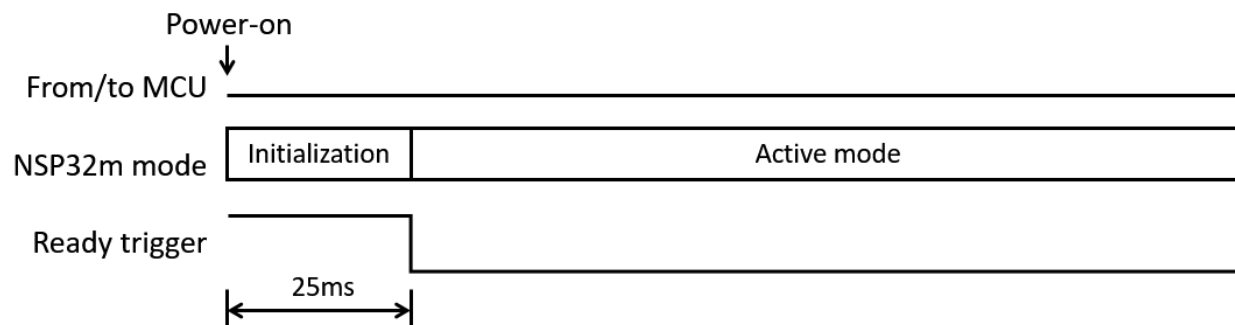
X				Y				Z				CHECKSUM
0x00	0x00	0x48	0x42	0x00	0x00	0xC8	0x42	0x00	0x00	0xB4	0x42	0x6D

Fixed value  
 Variable value

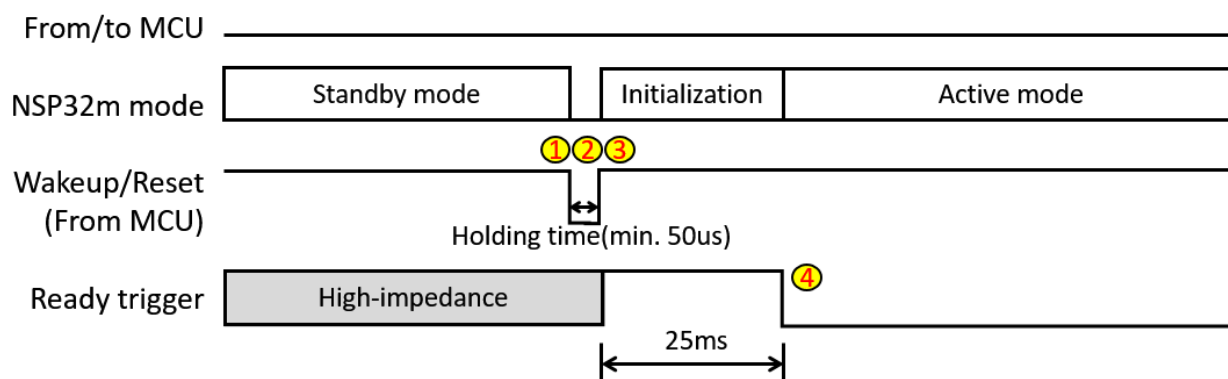
- Command 'CMD\_GET\_XYZ' is used for retrieving the measured tristimulus X, Y and Z values, initiated by "CMD\_ACQ\_XYZ". Please refer to section 3.4.7 for the description of [ IntegrationTime ], [ SaturationFlag ], [ X ], [ Y ], and [ Z ].

## 4. Timing Diagram

### 4.1 'Power on' sequence



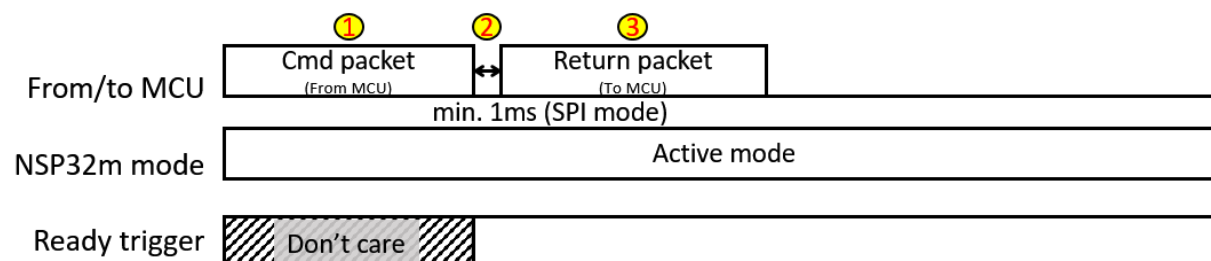
### 4.2 'Wakeup/Reset' sequence



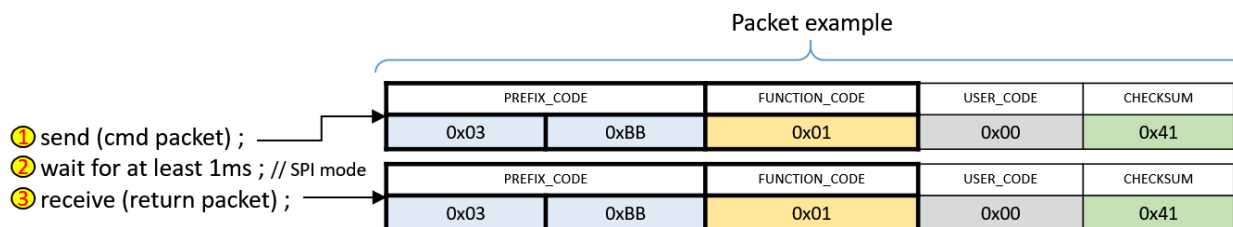
#### 'Wakeup/Reset' pseudo code

- ① set GPIO\_WAKEUP\_PIN to low ;
- ② wait for holding time (at least 50us) ;
- ③ set GPIO\_WAKEUP\_PIN to high ;
- ④ receive a 'Ready trigger' interrupt ; // which indicates the NSP32m is already entered active mode ;

### 4.3 'CMD\_HELLO' sequence

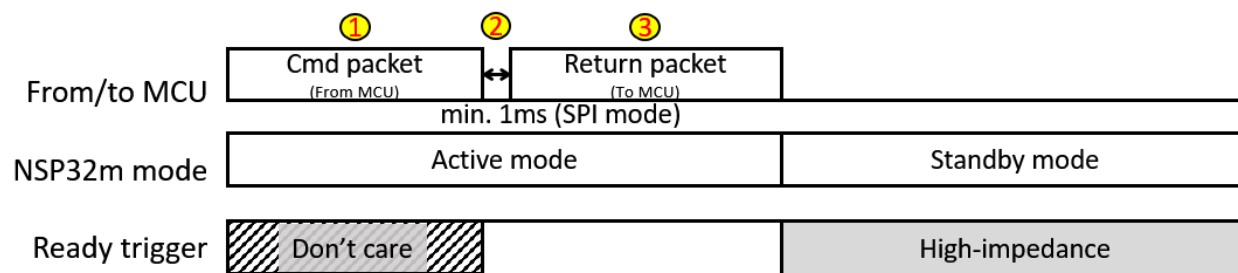


## 'CMD\_HELLO' pseudo code

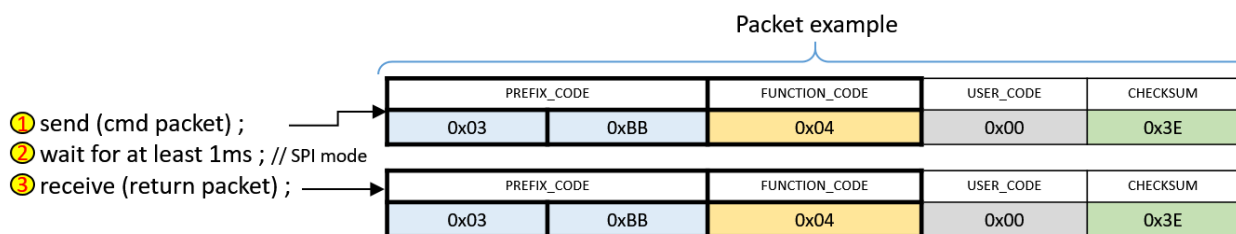


Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

## 4.4 'CMD\_STANDBY' sequence

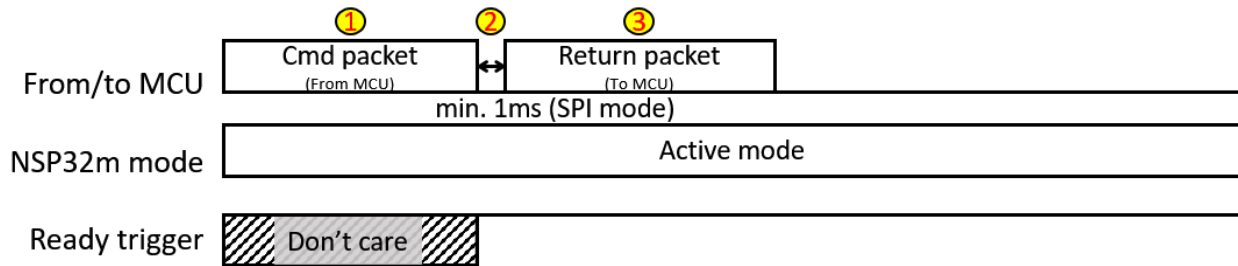


## 'CMD\_STANDBY' pseudo code

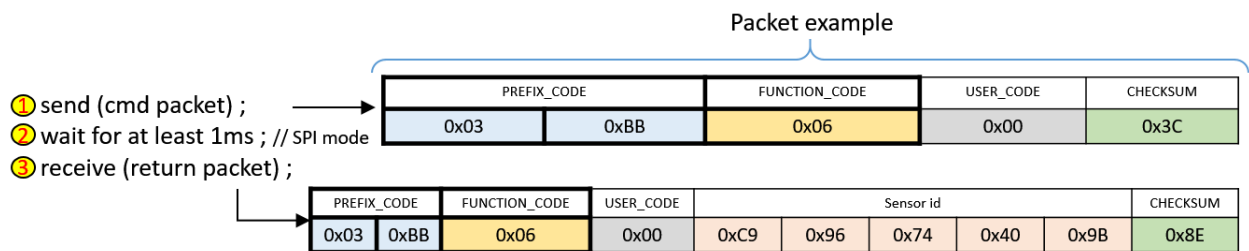


Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

## 4.5 'CMD\_GET\_SENSOR\_ID' sequence

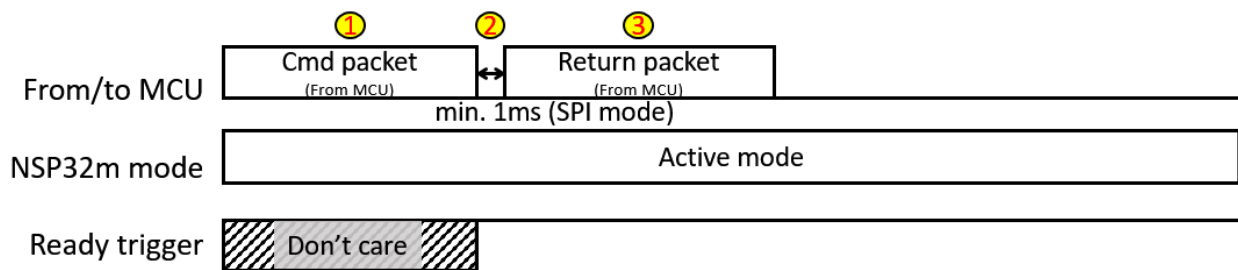


### 'CMD\_GET\_SENSOR\_ID' pseudo code

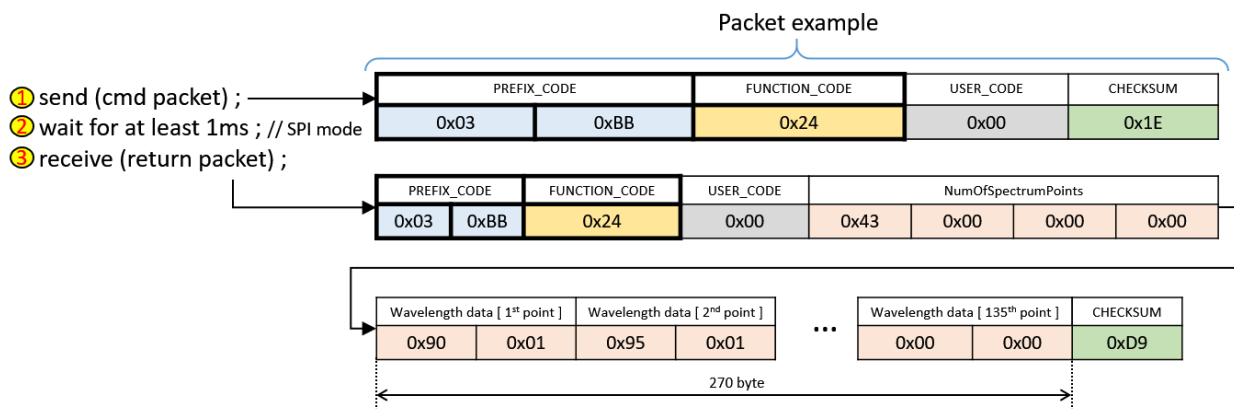


Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

## 4.6 'CMD\_GET\_WAVELENGTH' sequence

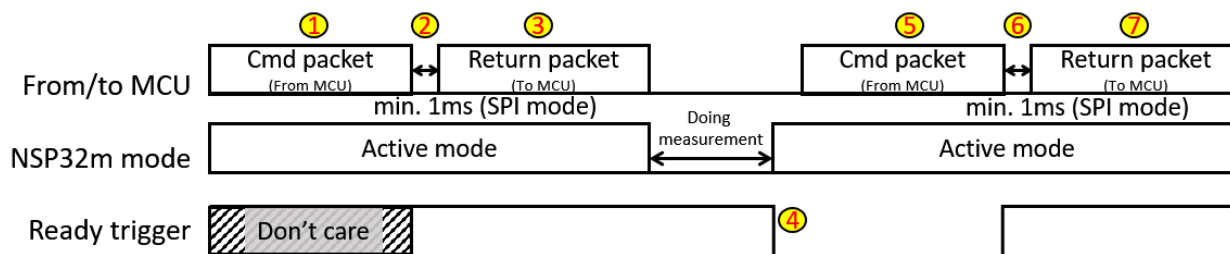


## 'CMD\_GET\_WAVELENGTH' pseudo code



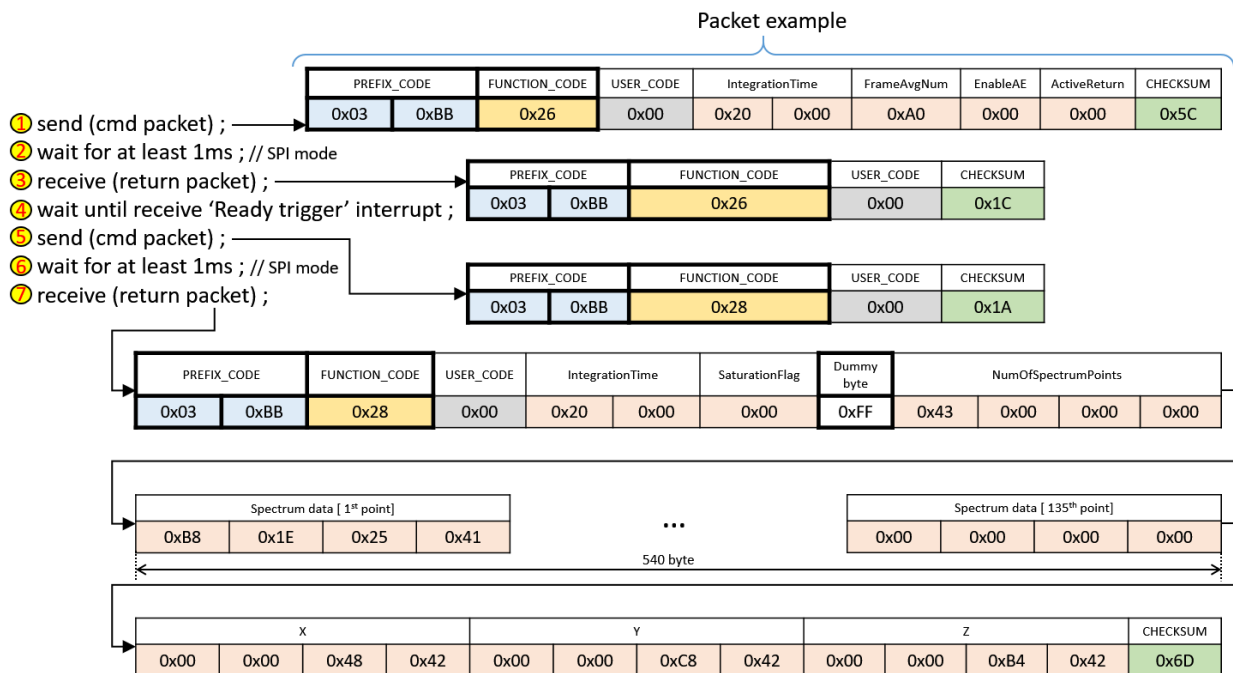
Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

## 4.7 'CMD\_ACQ\_SPECTRUM' and 'CMD\_GET\_SPECTRUM' sequence



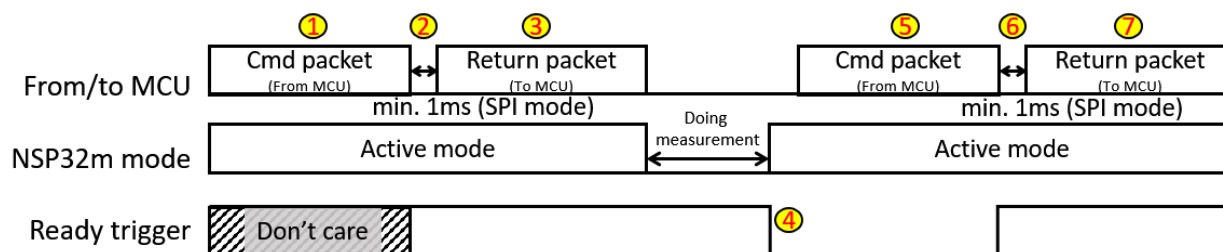


## 'CMD\_ACQ\_SPECTRUM' and 'CMD\_GET\_SPECTRUM' pseudo code

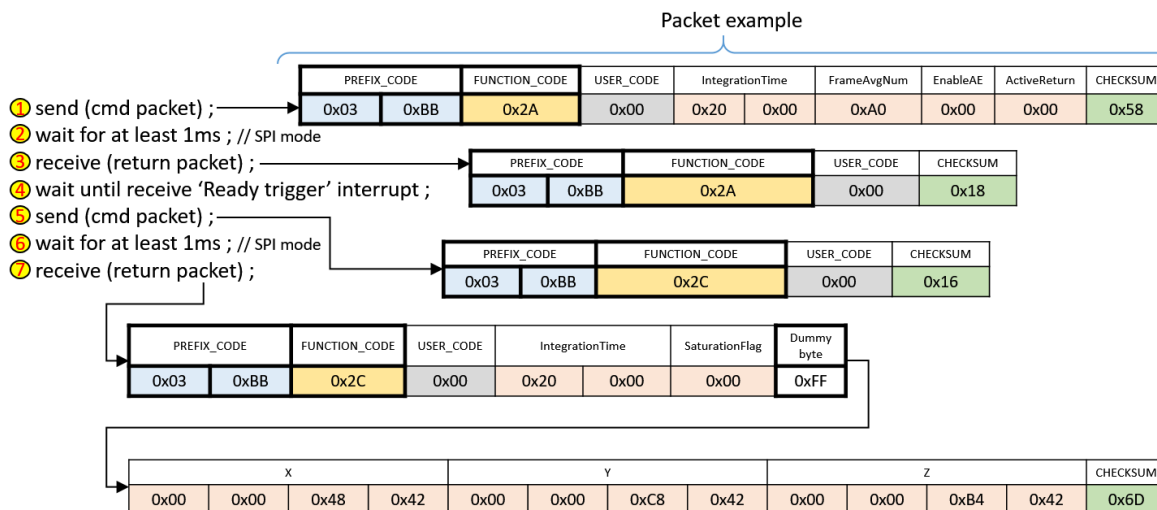


Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

### 4.8 'CMD\_ACQ\_XYZ' and 'CMD\_GET\_XYZ' sequence

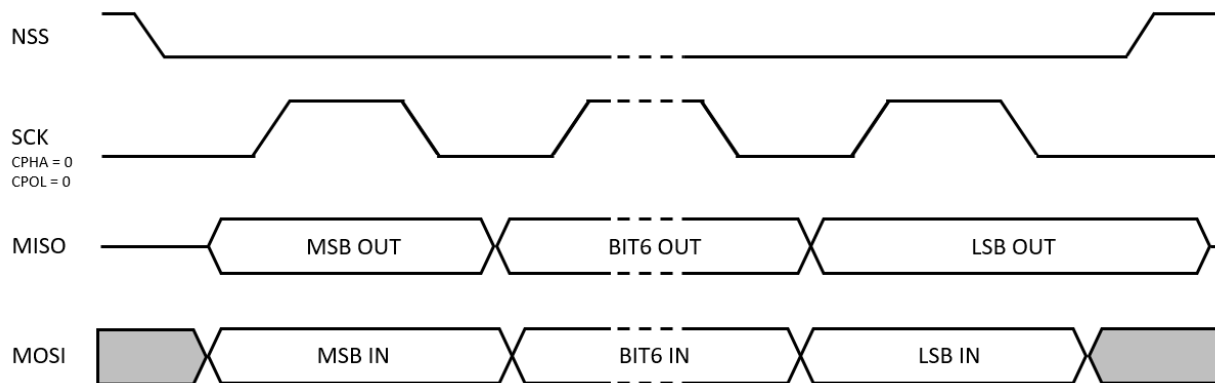


## 'CMD\_ACQ\_XYZ' and 'CMD\_GET\_XYZ' pseudo code



Note ) The 'return package' must be completely retrieved within 100ms, upon NSP32m completed receiving the 'cmd package'.

### 4.9 SPI Timing Diagram (Mode 0)



#### NOTE )

- SPI baud rate max. @4Mbits/sec for receiving return packet from NSP32m, and max. @ 2Mbits/sec for sending command packet to NSP32m.
- For more information about SPI timing diagram, please refer to <https://www.st.com/resource/en/datasheet/stm32f411ce.pdf>

## 5. Electrical Characteristics

### 5.1 SPI Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
SPI baud rate	Sending command packet to NSP32m	-	-	2	Mbits/s
	Receiving return packet from NSP32m	-	-	4	

Parameters	Description
Frame Format	Motorola
Data Size	8 bits
First Bit	MSB first
Mode	mode 0 (CPOL = 0; CPHA = 0)
CRC	None

### 5.2 UART Characteristics

Parameters	Description
Data bits	8 bits
Stop bit	1 bit
Parity	NONE
Flow control	NONE
Baud rate	115200bps, 38400bps, 19200bps, 9600bps, Selected by pins BAUD_SELECTION[1] BAUD_SELECTION[0]

### 5.3 Wakeup/Reset Pin Characteristics

Parameter	Min	Typ	Max	Unit
Weak pull-up equivalent resistor	30	40	50	k $\Omega$
Input low level voltage	-	-	0.3V <sub>DD</sub>	V
Input high level voltage	0.7 V <sub>DD</sub>	-	-	V
Holding time	50	-	-	us

### 5.4 Ready Trigger Pin Characteristics

Parameter	Min	Typ	Max	Unit
Output current	-	-	25	mA
Output low level voltage	-	-	0.4	V
Output high level voltage	2.4	-	-	V
Output fall time & rise time	-	-	100	ns

### 5.5 General Characteristics



Parameter	Min	Typ	Max	Unit
Power supply range	3.0	3.3	3.6	V

Power consumption	Active mode	-	-	23	mA
	Rebooting			23	mA
	Standby mode	-	-	11	uA
Rebooting time				25	ms
Temperature range		0	-	60	°C

## 6. Platforms Supported from NSP32m

NSP32m supports a wide range of platforms via SPI or UART interfaces. The table below summaries, but not limited to, the available libraries and examples provided by nanolambda. For further information and example source codes, please check [www.nanolambda.com](http://www.nanolambda.com).

		C/C++	C#	Java	Python	
API Library and Sample Code		<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
API Library		✓ <a href="#">(PDF)</a>	✓ <a href="#">(PDF)</a>	✓ <a href="#">(PDF)</a>	✓ <a href="#">(PDF)</a>	✓ <a href="#">(PDF)</a>
MCU	Arduino Example	✓ <a href="#">(PDF)</a> 1)Beginner <sup>[1]</sup> 2)Console Demo <sup>[1][3]</sup>				
	nRF52832 Example	✓ <a href="#">(PDF)</a> 1)BLE Demo <sup>[1][2]</sup> (in conjunction with Android GUI example)				
Raspberry Pi (Raspbian) Example					✓ <a href="#">(PDF)</a> 1)Beginner <sup>[1][3]</sup> 2)Console Demo <sup>[1][3]</sup> 3)Spectrum Meter <sup>[1][3]</sup>	
Android Example				✓ <a href="#">(PDF)</a> 1) GUI Demo <sup>[2]</sup> (in conjunction with nRF52832 BLE example)		
Windows Example			✓ <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>	✓ <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>		✓ <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>
Ubuntu Example				✓ <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>		✓ <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>

<b>macOS Example</b>			 <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>		 <a href="#">(PDF)</a> 1)Beginner <sup>[3]</sup> 2)Console Demo <sup>[3]</sup> 3)Spectrum Meter <sup>[3]</sup>
----------------------	--	--	--	--	--

<sup>[1]</sup> Examples of using SPI connection

<sup>[2]</sup> Examples of using Bluetooth connection via nRF52832

<sup>[3]</sup> Examples of using UART connection (an additional USB-to-UART adopter might be required)

## REVISION HISTORY

REVISION	DATE	REMARK
1.0.0	December 25, 2018	1 <sup>st</sup> Draft

## IMPORTANT NOTICE

nanoLambda Korea and its affiliates (“nanoLambda”) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to nanoLambda’s terms and conditions of sale supplied at the time of order acknowledgment. Customers are responsible for their products and applications using any nanoLambda products. nanoLambda does not warrant or represent that any license, either express or implied, is granted under any nanoLambda patent right, copyright, mask work right, or other nanoLambda intellectual property right relating to any combination, machine, or process in which nanoLambda products or services are used. Information published by nanoLambda regarding third-party products or services does not constitute a license from nanoLambda to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from nanoLambda under the patents or other intellectual property of nanoLambda. Reproduction of nanoLambda information in nanoLambda documents or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. nanoLambda is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions. Resale of nanoLambda products is not allowed without written agreement. Decompiling, disassembling, reverse engineering or attempt to reconstruct, identify or discover any source code, underlying ideas, techniques or algorithms are not allowed by any means. nanoLambda products are not authorized for use in safety-critical applications. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their

---

applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of nanoLambda products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by nanoLambda. Further, buyers must fully indemnify nanoLambda and its representatives against any damages arising out of the use of nanoLambda products in such safety-critical applications. This Notice shall be governed by and construed in accordance with the laws of Korea, without reference to principles of conflict of laws or choice of laws. All controversies and disputes arising out of or relating to this Notice shall be submitted to the exclusive jurisdiction of the Daejeon District Court in Korea as the court of first instance.