Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

UM1017
User manual

STEVAL-MKI062V2 communication protocol

Introduction

The scope of this user manual is to present the communication protocol used between the STEVAL-MKI062V2 demonstration board (iNEMO™ V2) and the iNEMO SDK (software development kit). This communication protocol runs on a physical communication channel based on USB virtual COM, which represents the physical channel used in the communication between the STEVAL-MKI062V2 and the PC.

The first chapter explains the general frame format and main rules used in the protocol.

The second chapter explains all the frames used in the actual release of the embedded firmware and software development kit (SDK).

Contents

List of tables

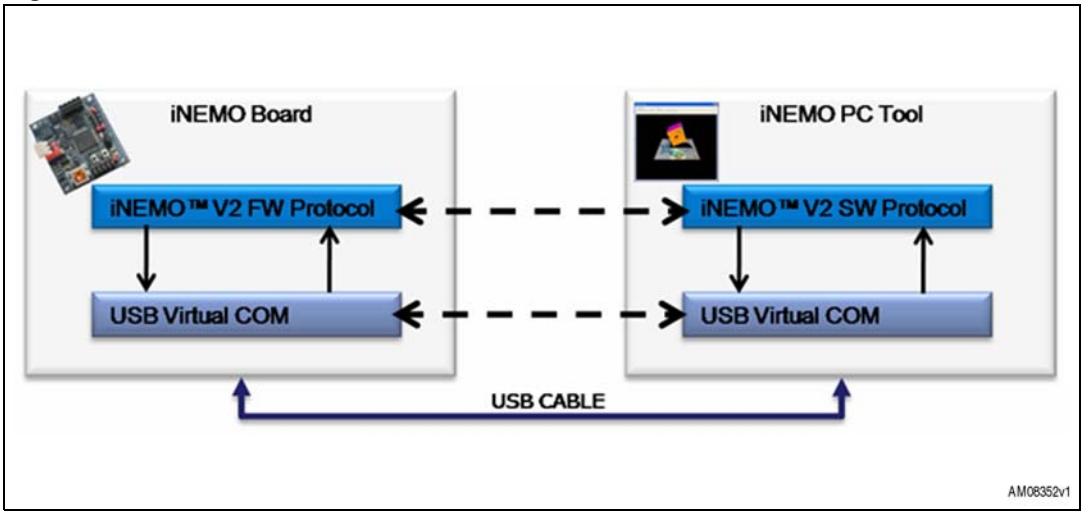
List of figures

1 General frame format and protocol rules

1.1 Frame format

This section explains the format of the frame used in the STEVAL-MKI062V2 communication protocol. The STEVAL-MKI062V2 exchanges data and commands with the PC GUI through a physical communication channel based on USB virtual COM. Each frame, described below, represents the payload of a USB frame.

Figure 1. STEVAL-MKI062V2 communication architecture



The frames are described as a sequence of fields in a specific order. All frame formats are depicted in the order in which they are passed to the USB driver, from left to right. Bits within each field are numbered from k-1 (leftmost and most significant) to 0 (rightmost and least significant), where the length of the field is k bits.

The frame format is composed of a header and an optional payload. The general frame is formatted as illustrated in *Figure 2*. The header is composed of three mandatory (M) fields, each of which is 1 byte in length, while the payload is an optional field whose maximum length is 61 bytes. See LF/MF field in the following section to exceed this limit.

Figure 2. General frame format

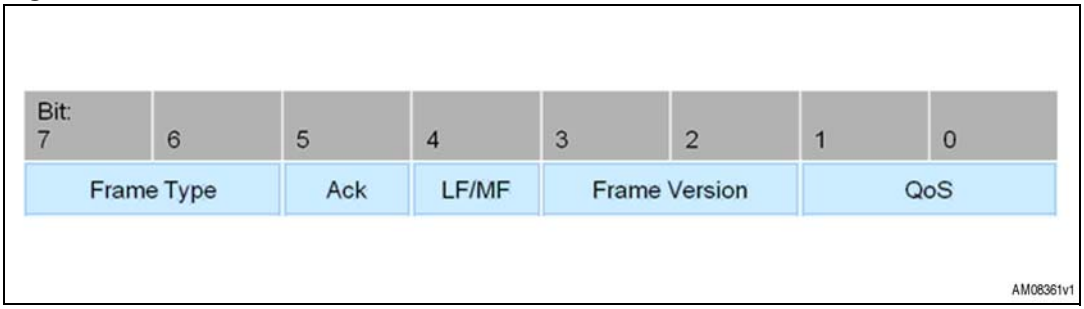
Bytes: 1	1	1	≤ 61
Frame Control (M)	Length (M)	Message ID (M)	Payload (O)

AM08360v1

1.1.1 Frame control field

The frame control field is 1 byte in length and contains information defining the frame type and other control flags. The frame control field is formatted as illustrated in *Figure 3*.

Figure 3. Frame control field



The frame type subfield is 2 bits in length and is set to one of the values listed in *Table 1*.

Table 1. Frame type list

Value	Frame type
00	CONTROL
01	DATA
10	ACK
11	NACK

The ACK subfield is 1 bit in length and specifies whether an acknowledgement is required from the recipient on receipt of a DATA or CONTROL frame. If this field is set to one, the recipient sends an acknowledgment frame only if, upon reception, the frame passes all required levels of filtering. If this subfield is set to zero, the recipient device does not send an acknowledgment frame. It is possible to embed a payload in an acknowledgment frame (piggybacking) to send useful information to the transmitter and avoid further transactions. When the ACK field is set to one, and if, upon reception the frame doesn't pass the required level of filtering, the recipient sends a no-acknowledgment frame (NACK), whose payload is an error code (e.g. unsupported command, value out of range,…). In the ACK and/or NACK frames the ACK field is set to zero and ignored upon reception.

The LF/MF (last fragment / more fragment) subfield is 1 bit in length and it is used for fragmentation and reassembling. This field is set to zero to indicate a single frame or the last frame of a multiple-frame transaction. This field is set to 1 to indicate that other frames follow, all belonging to the same transaction. In the ACK and NACK frames (with or without payload) fragmentation is not supported and this subfield is set to zero in the transmission of ACK and NACK frames and ignored upon reception.

The frame version subfield is 2 bits in length and is set to “00” at this time. Values concerning future versions are “reserved for future use” (RFU) as listed in *Table 2*.

Table 2. Frame version list

Value	Frame version
00	Version 1.0
01	RFU
10	
11	

The QoS (Quality of Service) subfield is 2 bits in length and is set to one of the values listed in *Table 3*. This subfield allows the application to exchange and process data and control frames with different priorities.

Table 3. QoS list

Value	QoS
00	Normal priority
01	Medium priority
10	High priority
11	RFU

1.1.2 Length field

The length field is 1 byte in length and contains the number of bytes that follow. Admitted values are in the range 1 to 62.

1.1.3 Message ID field

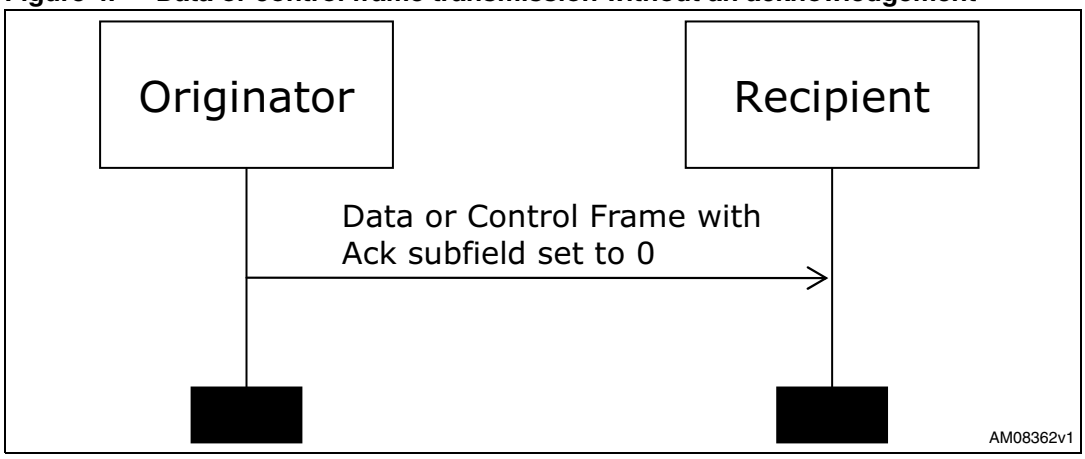
The message ID is 1 byte in length and contains an identifier of the user application messages. See *Section 2.2* and the following for further details.

1.2 Protocol rules

There are two types of transactions, according to whether the DATA or CONTROL frame is acknowledged or not.

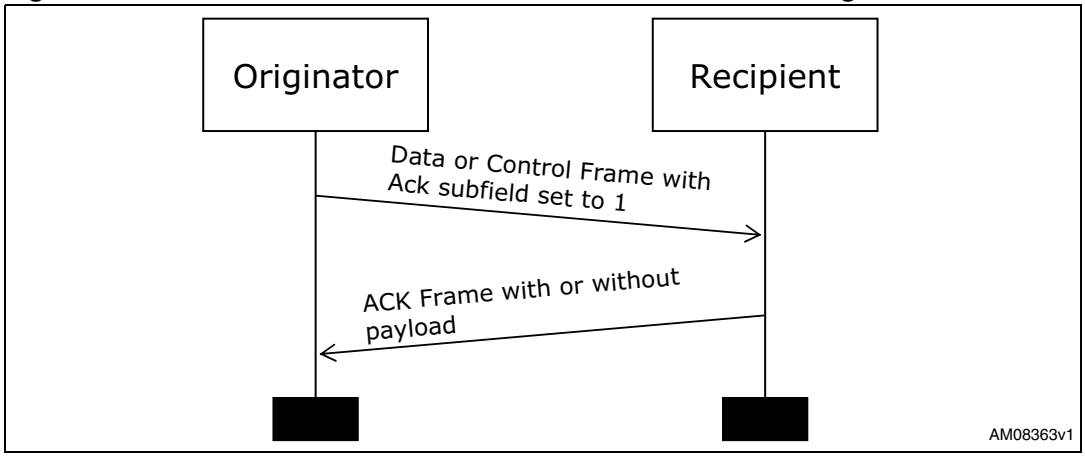
A DATA or CONTROL frame with the ACK subfield of its frame control field set to zero is not acknowledged by its intended recipient. The originating device (PC or iNEMO board) assumes that the transmission of the frame was successful. The message sequence chart in *Figure 4* shows the scenario for transmitting a single DATA or CONTROL frame from an originator to a recipient without requiring an acknowledgement.

Figure 4. Data or control frame transmission without an acknowledgement



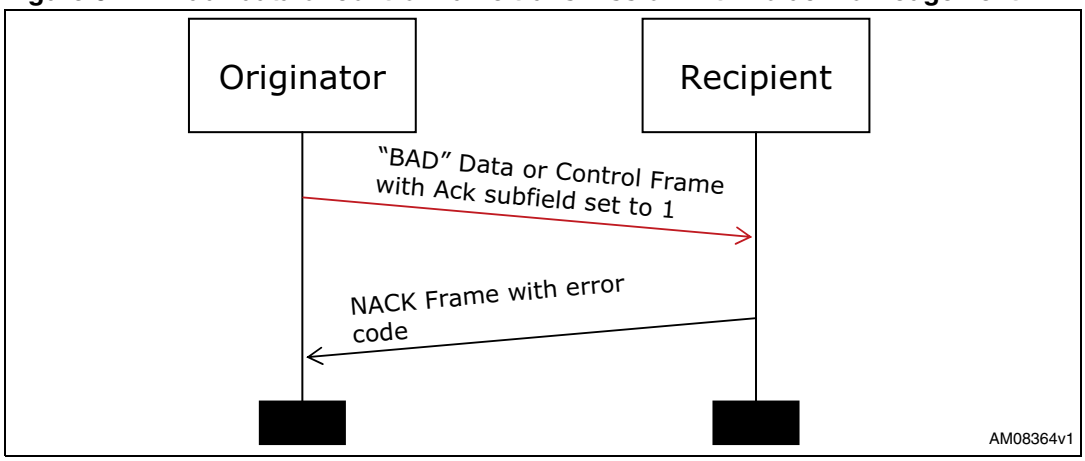
A DATA or CONTROL frame transmitted with the ACK subfield of its frame control field set to one is acknowledged by the recipient. If the intended recipient correctly receives the frame, it generates and sends an ACK frame containing the same message ID from the DATA or CONTROL frame that is being acknowledged. It is also possible to include a payload in the ACK frame to transfer useful data from the recipient to the originator. The message sequence chart in *Figure 5* shows the scenario for transmitting a single DATA or CONTROL frame from an originator to a recipient with an acknowledgement.

Figure 5. Data or control frame transmission with an acknowledgement



If the frame received does not pass all the required filtering rules, the recipient generates and sends a NACK frame containing the same message ID from the DATA or CONTROL frame that is being acknowledged and contains the error code. The message sequence chart in *Figure 6* shows the scenario for transmitting a single "bad" DATA or CONTROL frame from an originator to a recipient with a no-acknowledgement.

Figure 6. "Bad" data or control frame transmission with no-acknowledgement



2 STEVAL-MKI062V2 frames

2.1 STEVAL-MKI062V2 frame types

The frames used in the STEVAL-MKI062V2 are classified in four types:

1. Communication control frames
2. Board information frames
3. Sensor setting frames
4. Acquisition sensor data frames

2.2 Communication control frames

Communication control frames are frames originated by the software PC (SDK or GUI) and used to send specific commands to the iNEMO board. All the communication control frames are listed in *Table 4*.

Table 4. Communication control frames

Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Connect	CONTROL	Y	0x00	N	0		PC
iNEMO_Connect_Response	ACK	N	0x00	N	0		iNEMO
	NACK	N	0x00	N	1	Error code	
iNEMO_Disconnect	CONTROL	Y	0x01	N	0		PC
iNEMO_Disconnect_Response	ACK	N	0x01	N	0		iNEMO
	NACK	N	0x01	N	1	Error code	
iNEMO_Reset_Board	CONTROL	Y	0x02	N	0		PC
iNEMO_Reset_Board_Response	ACK	N	0x02	N	0		iNEMO
	NACK	N	0x02	N	1	Error code	
iNEMO_Enter_DFU_Mode	CONTROL	Y	0x03	N	0		PC
iNEMO_Enter_DFU_Mode_Response	ACK	N	0x03	N	0		iNEMO
	NACK	N	0x03	N	1	Error code	
iNEMO_Trace	CONTROL	Y	0x07	N	0		PC
iNEMO_Trace_Response	ACK	N	0x07	N	0		iNEMO
	NACK	N	0x07	N	1	Error code	

Table 4. Communication control frames (continued)

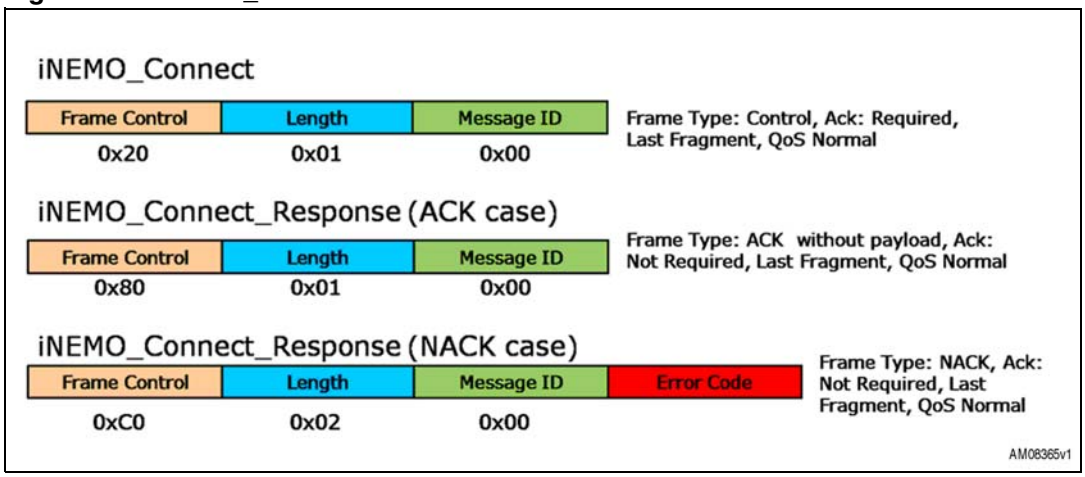
Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Trace_Data	DATA	N	0x07	M	Variable	String for debug purpose	
iNEMO_Led_Control	CONTROL	Y	0x08	N	1	0x00 OFF 0x01 ON	PC
iNEMO_Led_Control_Response	ACK	N	0x08	N	0		iNEMO
	NACK	N	0x08	N	1	Error code	

2.2.1 iNEMO_Connect

The iNEMO_Connect command is the first command sent from the GUI or SDK to the iNEMO board. Any other command sent before the iNEMO_Connect will not be processed by iNEMO. It works like a "ping" and opens the communication between the GUI or SDK and the iNEMO board at the application level.

Figure 7 shows the frames involved in the iNEMO_Connect transaction.

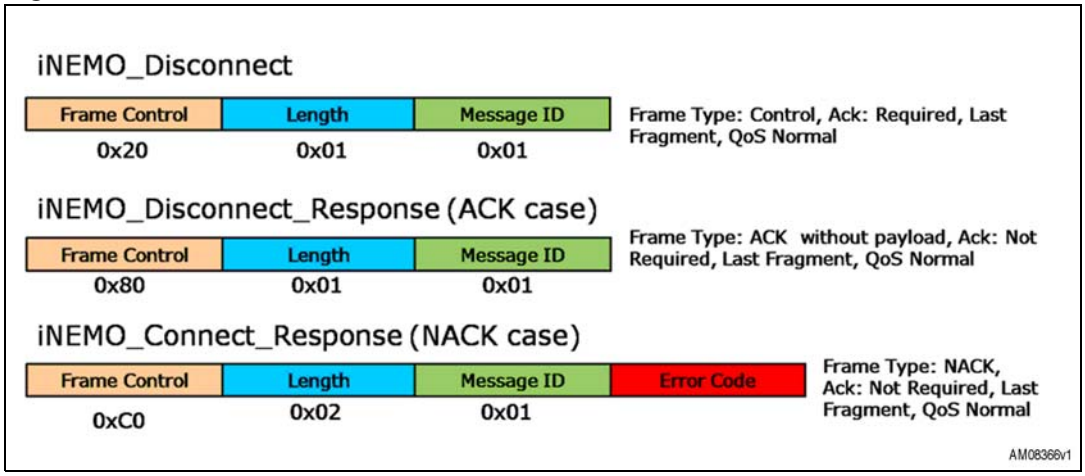
Figure 7. iNEMO_Connect frames



2.2.2 iNEMO_Disconnect

The iNEMO_Disconnect command closes the communication between the PC and the iNEMO board. *Figure 8* shows the frames involved in the iNEMO_Disconnect transaction.

Figure 8. iNEMO_Disconnect frames



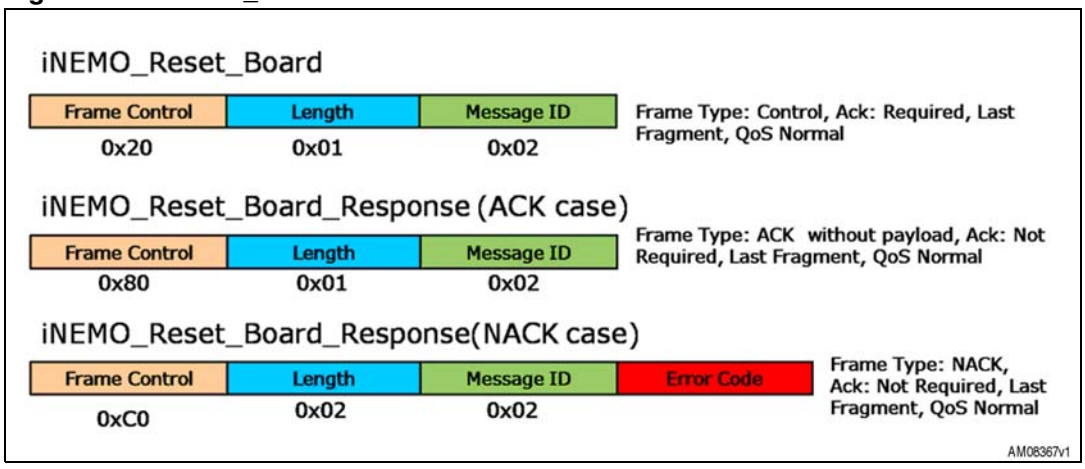
The GUI (or SDK), after receiving the ACK frame, closes the USB Virtual Com. To re-open the communication, use only the iNEMO_Connect command.

2.2.3 iNEMO_Reset

The iNEMO_Reset command initiates a software reset of the iNEMO board. After receiving the iNEMO_Reset command, the iNEMO board replies with the ACK frame, then waits for 5 seconds before disconnecting the USB cable in the software and invokes a software reset. The GUI (or SDK), after receiving the ACK frame, closes the USB Virtual Com. To re-open the communication, use only the iNEMO_Connect command.

Figure 9 shows the frames involved in the iNEMO_Reset transaction.

Figure 9. iNEMO_Reset frames

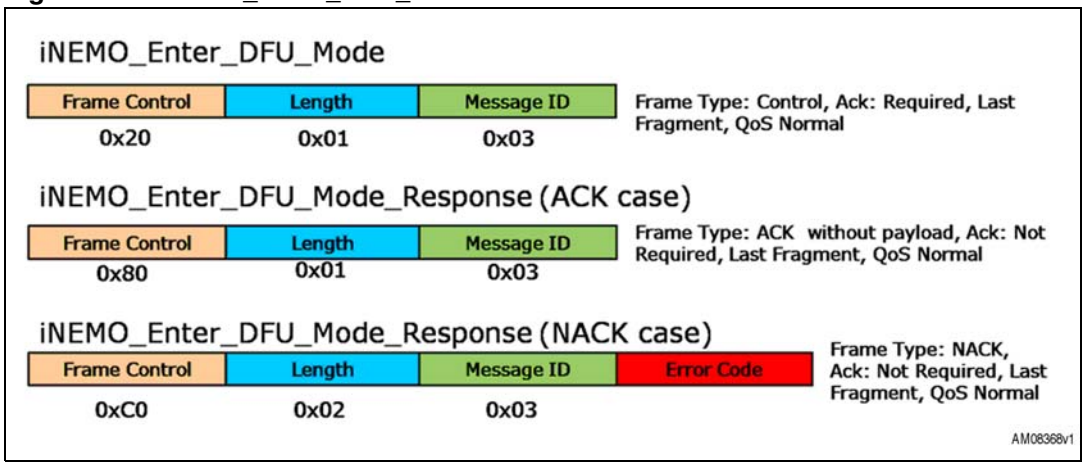


2.2.4 iNEMO_Enter_DFU_Mode

The iNEMO_Enter_DFU_Mode command allows the iNEMO board to enter in DFU mode in the software mode. After receiving the iNEMO_Enter_DFU_Mode command, the iNEMO board replies with an ACK frame. Then it sets the Option Byte Data0 (at address 0x1FFFF804) to one, disconnects the USB cable in the software and it invokes a software reset. After reset, iNEMO enters in DFU mode. After entering in DFU mode in the software, iNEMO changes the option byte Data0 to zero. The user can leave the DFU mode in two ways: un-plugging and plugging the USB cable (hardware mode), or using the Leave_DFU_Mode command available in the DfuSe demo PC application or in the GUI or SDK. The GUI (or SDK) closes the USB Virtual Com after receiving the ACK frame.

Figure 10 shows the frames involved in the iNEMO_Enter_DFU_Mode transaction.

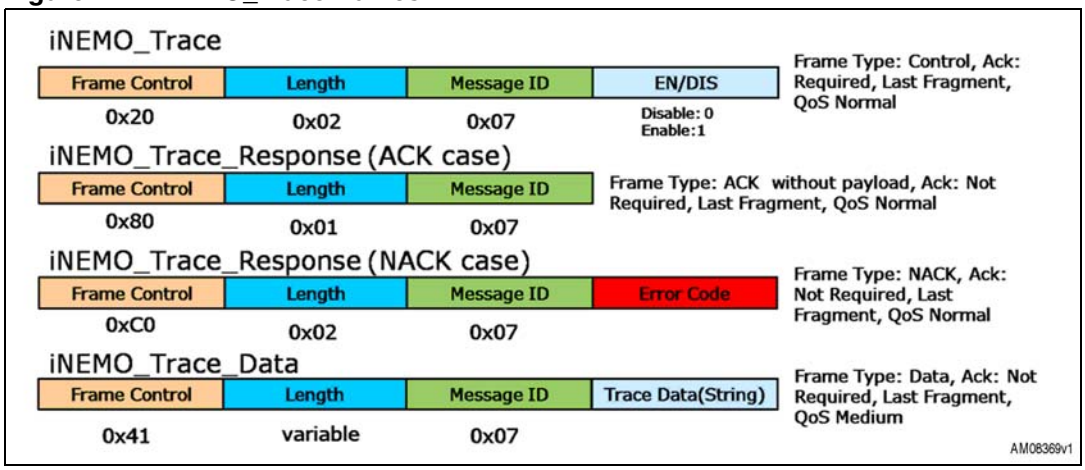
Figure 10. iNEMO_Enter_DFU_Mode frames



2.2.5 iNEMO_Trace

The iNEMO_Trace command allows the user to enable or disable "trace data". Trace data are used for debugging purposes and they are displayed as a string in a debug window. The frames are asynchronous and have medium priority (QoS sub-field of frame control field). *Figure 11* shows the frames involved in the iNEMO_Trace transaction.

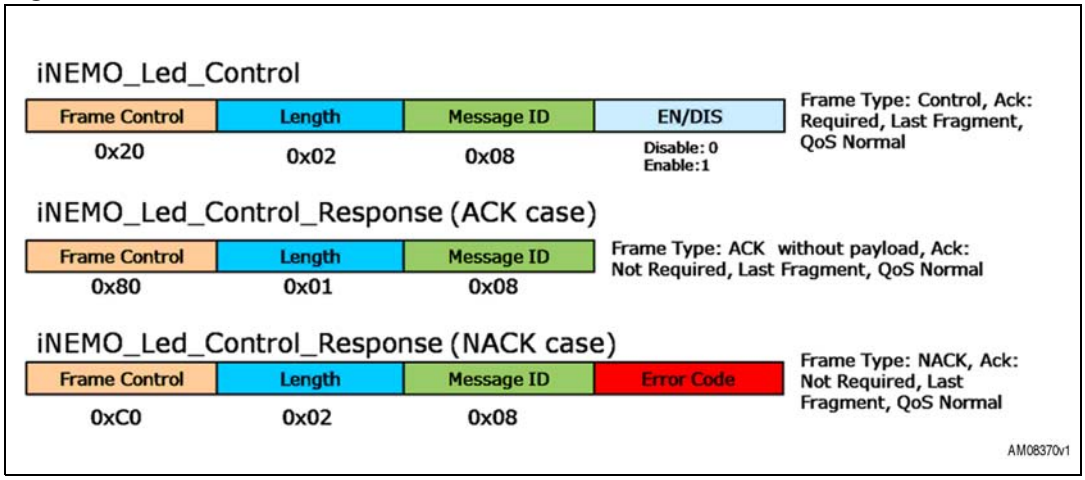
Figure 11. iNEMO_Trace frames



2.2.6 iNEMO_Led_Control

The iNEMO_Led_Control command allows turning on or off the LED available on the iNEMO board. *Figure 12* shows the frames involved in the iNEMO_Led_Control transaction.

Figure 12. iNEMO_Led_Control frames



2.3 Board information frames

Board information frames are frames originated by the software PC (SDK or GUI) and used to retrieve information about firmware or the hardware features of the iNEMO board. All the board information frames are listed in *Table 5*.

Table 5. Board information frames

Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Get_Device_Mode	CONTROL	Y	0x10	N	0		PC
iNEMO_Get_Device_Mode_Response	ACK	N	0x10	N	1	0x00 sensor mode 0x01 master mode	iNEMO
	NACK	N	0x10	N	1	Error code	
iNEMO_Get_MCU_ID	CONTROL	Y	0x12	N	0		PC
iNEMO_Get_MCU_ID_Response	ACK	N	0x12	N	12	Unique device ID	iNEMO
	NACK	N	0x12	N	1	Error code	
iNEMO_Get_FW_Version	CONTROL	Y	0x13	N	0		PC
iNEMO_Get_FW_Version_Response	ACK	N	0x13	N	Variable	String firmware version	iNEMO
	NACK	N	0x13	N	1	Error code	

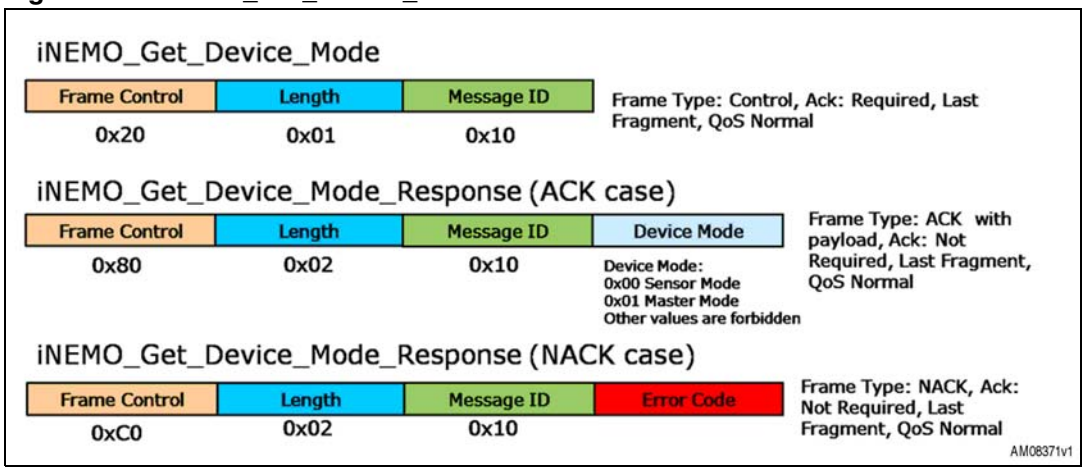
Table 5. Board information frames (continued)

Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Get_HW_Version	CONTROL	Y	0x14	N	0	Date, time	PC
iNEMO_Get_HW_Version_Response	ACK	N	0x14	N	Variable	String hardware version	iNEMO
	NACK	N	0x14	N	1	Error code	
iNEMO_Identify	CONTROL	Y	0x15	N	0		PC
iNEMO_Identify_Response	ACK	N	0x15	N	12	Unique device ID	iNEMO
	NACK	N	0x15	N	1	Error code	
iNEMO_Get_AHRS_Library	CONTROL	Y	0x17	N	0		PC
iNEMO_Get_AHRS_Library_Response	ACK	N	0x17	N	Variable	AHRS enable/disable string	iNEMO
	NACK	N	0x17	N	1	Error code	
iNEMO_Get_Libraries	CONTROL	Y	0x18	N	0		PC
iNEMO_Get_Libraries_Response	ACK	N	0x18	N	0	List of supported libraries	iNEMO
	NACK	N	0x18	N	1	Error code	

2.3.1 iNEMO_Get_Device_Mode

The iNEMO_Get_Device_Mode command allows knowing if the iNEMO board is working in master mode or in sensor mode. A device responding to be in master mode is responsible for managing a network of sensor nodes. It is physically connected to the PC and has the responsibility to retransmit commands coming from PC to the sensor node available in the network, as well as to retransmit data coming from sensor nodes available in the network to the PC. The default working model of iNEMO is sensor mode; master mode is not supported by iNEMO. *Figure 13* shows the frames involved in the iNEMO_Get_Device_Mode transaction.

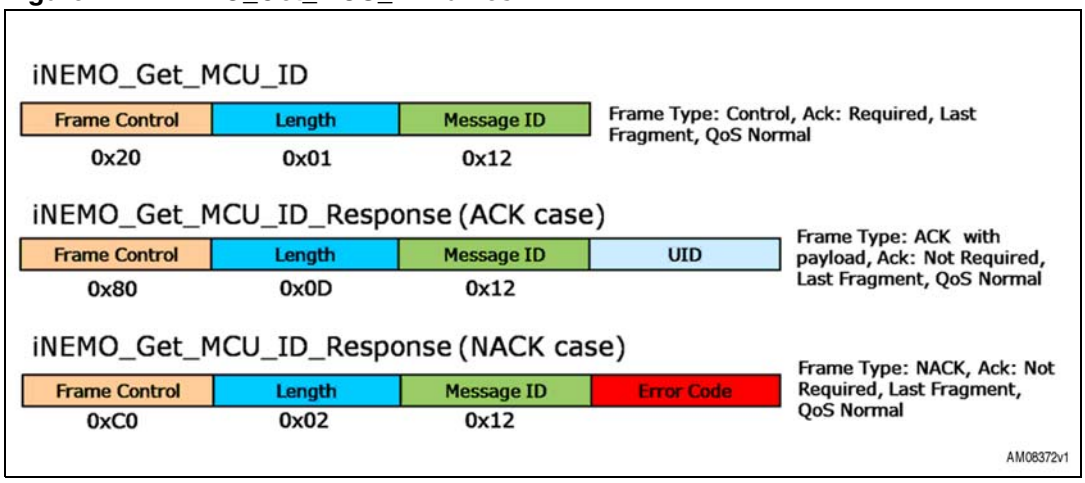
Figure 13. iNEMO_Get_Device_Mode frames



2.3.2 iNEMO_Get_MCU_ID

The iNEMO_Get_MCU_ID command allows retrieving from the iNEMO board the 96-bit unique device identifier of the STM32F103RE microcontroller [see http://www.st.com/stonline/products/literature/rm/13902.pdf for further details on this feature]. *Figure 14* shows the frames involved in the iNEMO_Get_MCU_ID transaction.

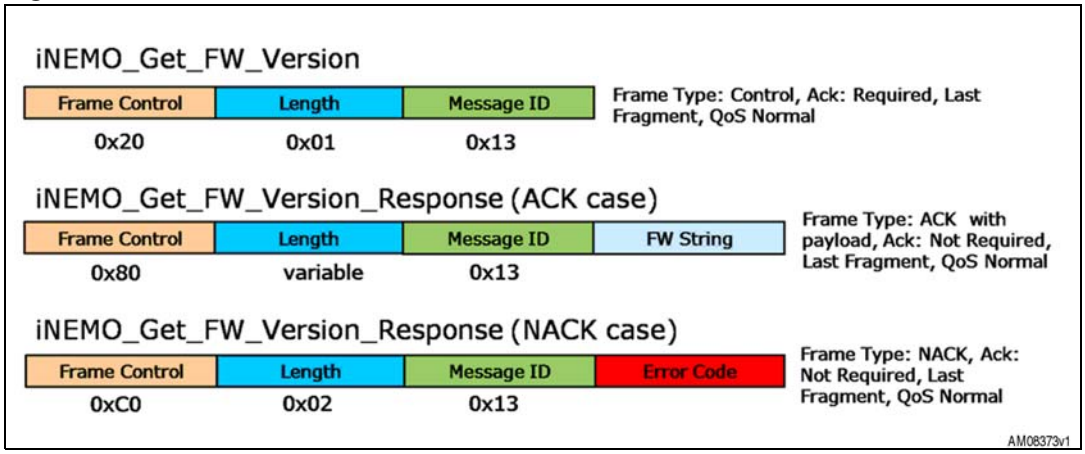
Figure 14. iNEMO_Get_MCU_ID frames



2.3.3 iNEMO_Get_FW_Version

The iNEMO_Get_FW_Version command allows retrieving the iNEMO firmware version. *Figure 15* shows the frames involved in the iNEMO_Get_FW_Version transaction.

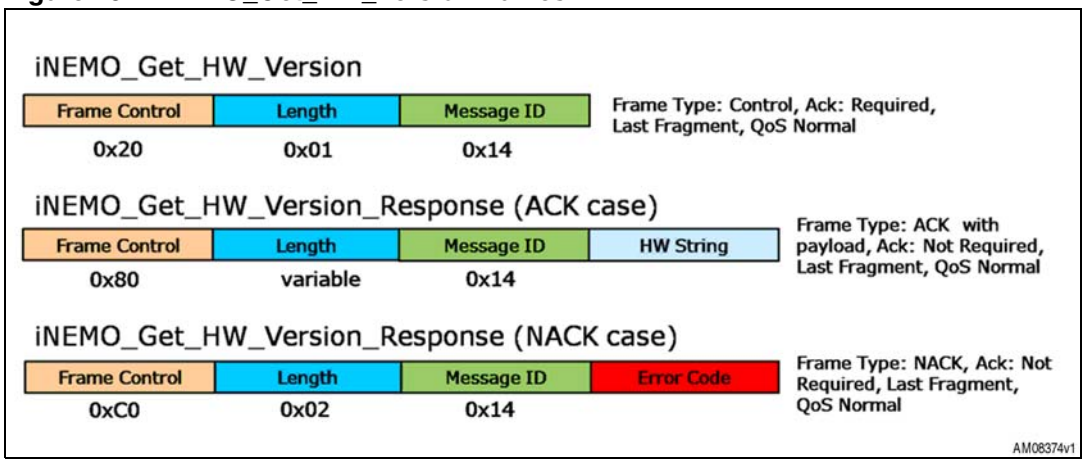
Figure 15. iNEMO_Get_FW_Version frames



2.3.4 iNEMO_Get_HW_Version

The iNEMO_Get_HW_Version command allows retrieving the iNEMO hardware version. *Figure 16* shows the frames involved in the iNEMO_Get_HW_Version transaction.

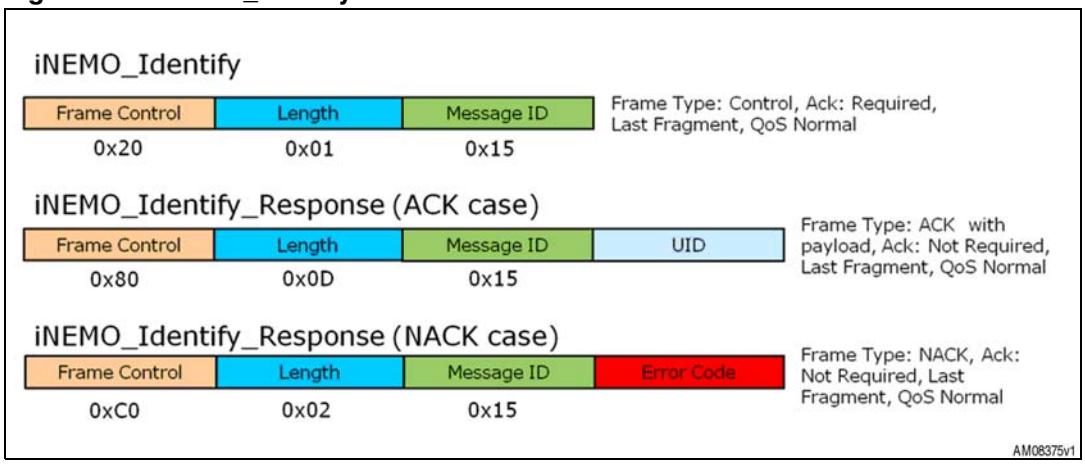
Figure 16. iNEMO_Get_HW_Version frames



2.3.5 iNEMO_Identify

The iNEMO_Identify command can be used to identify an iNEMO board. Upon reception of the iNEMO_Identify command, the iNEMO board replies with an ACK containing the MCU Unique Device ID. Then the LED available on the board blinks 3 times. *Figure 17* shows the frames involved in the iNEMO_Identify transaction.

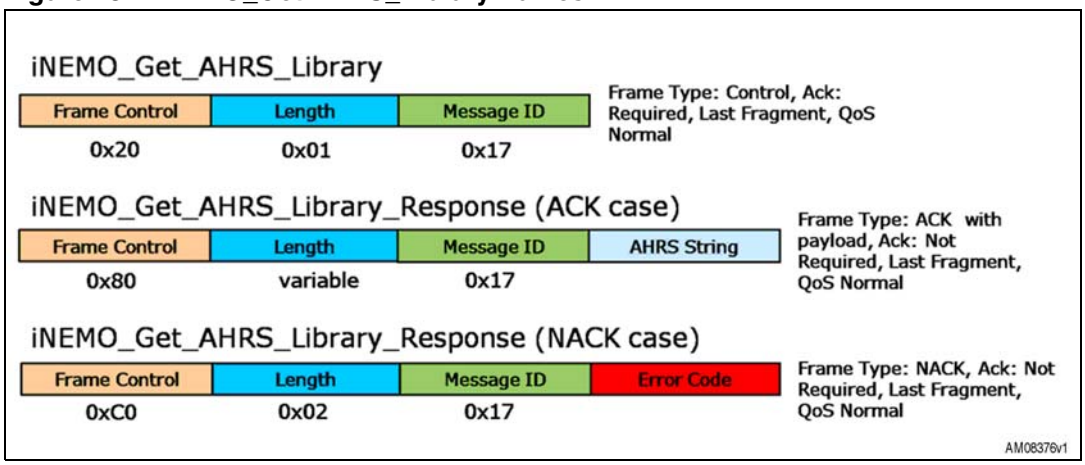
Figure 17. iNEMO_Identify frames



2.3.6 iNEMO_Get_AHRS_Library

The iNEMO_Get_AHRS_Library command allows knowing the version of the iNEMO firmware attitude heading reference system (AHRS) algorithm. The returned value is in string format. *Figure 18* shows the frames involved in the iNEMO_Get_AHRS_Library transaction.

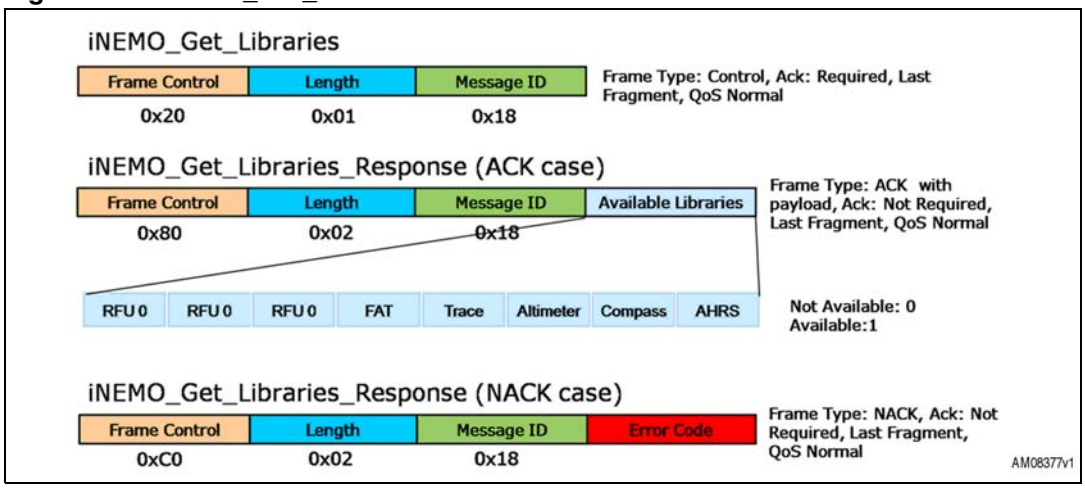
Figure 18. iNEMO_Get AHRS_Library frames



2.3.7 iNEMO_Get_Libraries

The iNEMO_Get_Libraries command allows knowing which specific libraries are supported by the iNEMO firmware. *Figure 19* shows the frames involved in the iNEMO_Get_Libraries transaction.

Figure 19. iNEMO_Get_Libraries frames



2.4 Sensor setting frames

Sensor setting frames are frames originated by the software PC (SDK or GUI) and used to set sensor parameters or to retrieve information about them. All the sensor setting frames are listed in *Table 6*.

Table 6. Sensor setting frames

Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Set_Sensor_Parameter	CONTROL	Y	0x20	N	variable	Sensor_Type, Sensor_Parameter, Parameter_Value	PC
iNEMO_Set_Sensor_Parameter_Response	ACK	N	0x20	N	0		iNEMO
	NACK	N	0x20	N	1	Error code	
iNEMO_Get_Sensor_Parameter	CONTROL	Y	0x21	N	2	Sensor_Type, Sensor_Parameter	PC
iNEMO_Get_Sensor_Parameter_Response	ACK	N	0x21	N	variable	Sensor_Type, Sensor_Parameter, Parameter_Value	iNEMO
	NACK	N	0x21	N	1	Error code	
iNEMO_Restore_Default_Parameter	CONTROL	Y	0x22	N	2	Sensor_Type, Sensor_Parameter	PC

Table 6. Sensor setting frames (continued)

Commands	Frame type	Ack required	Message ID	QoS	Payload length (in bytes)	Payload	Originator
iNEMO_Restore_Default_Parameter_Response	ACK	N	0x22	N	variable	Sensor_Type, Sensor_Parameter, Parameter_Value	iNEMO
	NACK	N	0x22	N	1	Error code	

2.4.1 iNEMO_Set_Sensor_Parameter

The iNEMO_Set_Sensor_Parameter command allows setting a specific sensor parameter. *Figure 20* shows the frames involved in the iNEMO_Set_Sensor_Parameter transaction.

Figure 20. iNEMO_Set_Sensor_Parameter frames

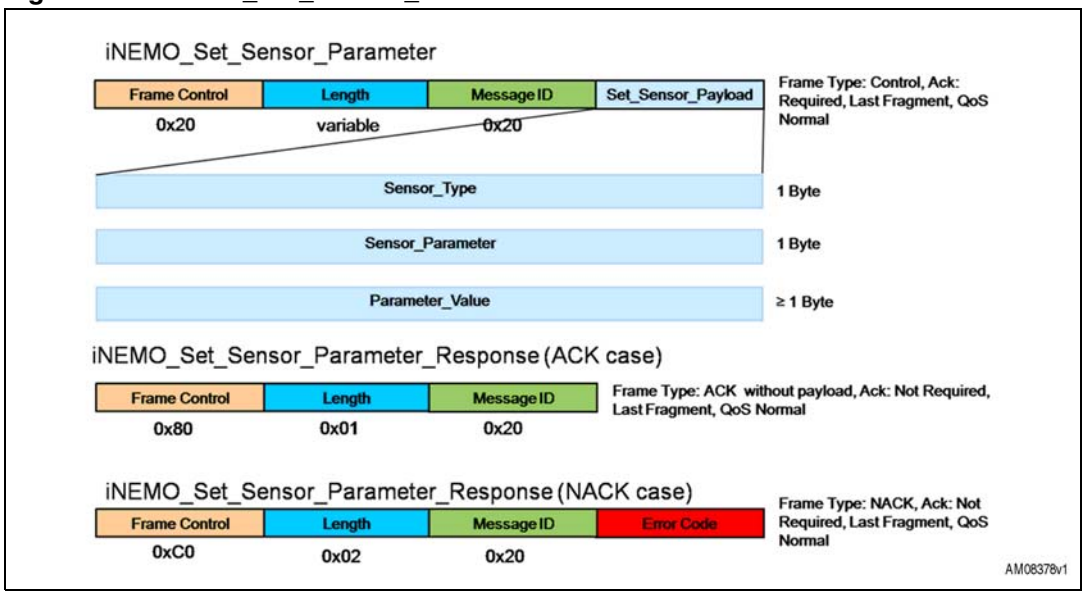


Table 7 lists the type of sensor and value for the "Sensor_Type" field.

2.4.2 iNEMO_Get_Sensor_Parameter

The iNEMO_Get_Sensor_Parameter command allows retrieving a specific sensor parameter from iNEMO . *Figure 21* shows the frames involved in the iNEMO_ Get_Sensor_Parameter transaction.

Figure 21. iNEMO_Get_Sensor_Parameter frames

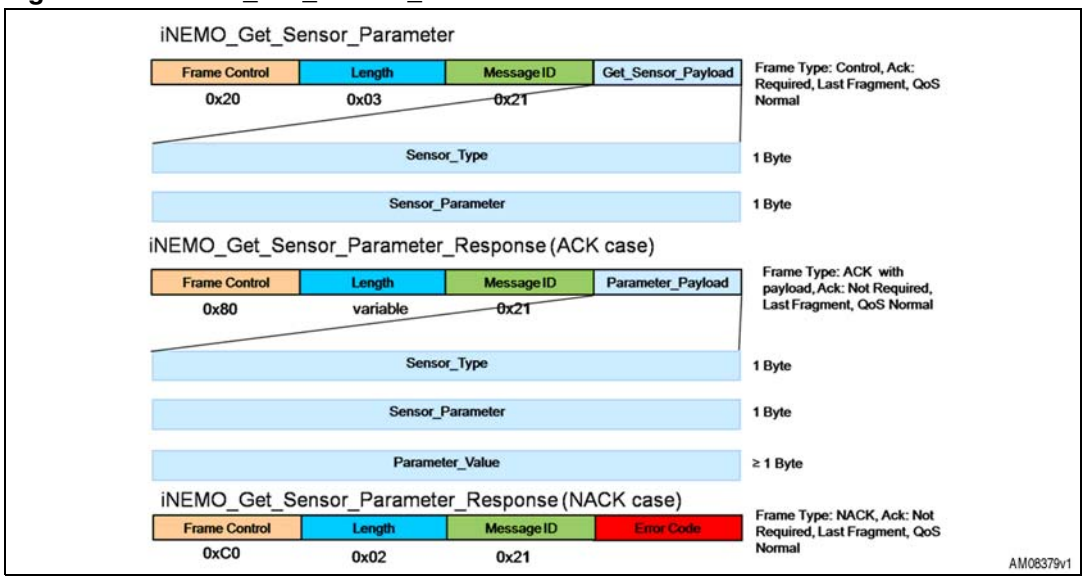


Table 7 lists the type of sensor and value for the "Sensor_Type" field.

2.4.3 iNEMO_Restore_Default_Parameter

The iNEMO_Restore_Default_Parameter command allows restoring a specific default sensor parameter. *Figure 22* shows the frames involved in the iNEMO_Restore_Default_Parameter transaction.

Figure 22. iNEMO_Restore_Default_Parameter frames

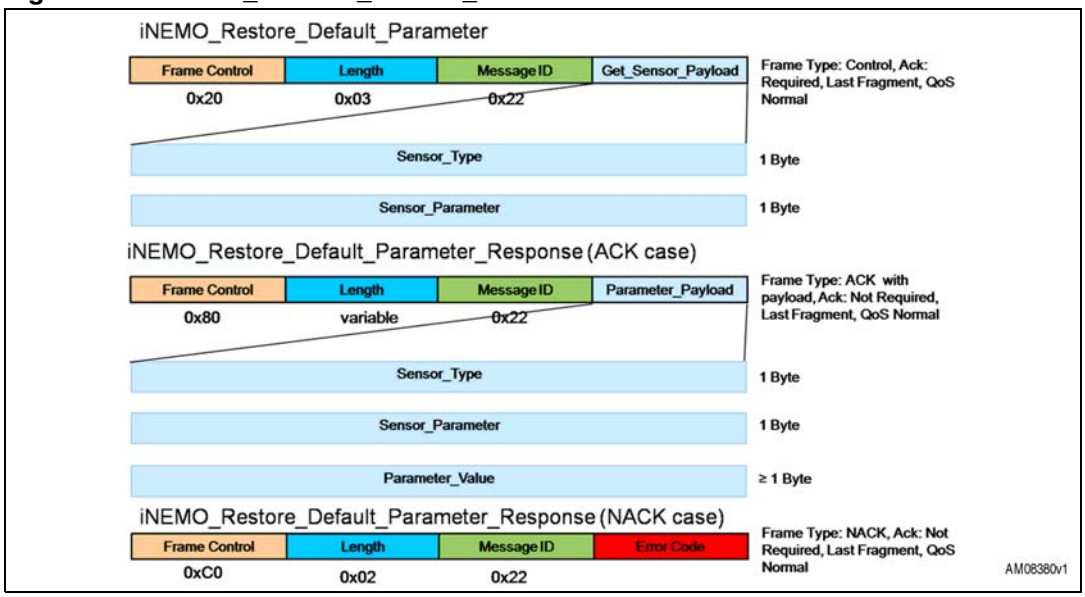


Table 7 lists the type of sensor and value for the "Sensor_Type" field.

Table 7. Sensor_Type field

Sensor_Type field	Sensor
0x00	3-axis accelerometer
0x01	3-axis magnetometer
0x02	2-axis gyroscope (pitch/roll)
0x03	1-axis gyroscope (Yaw)
0x04	Pressure
0x05	Temperature
0x06 – 0xFF	Reserved for future use

2.4.4 Accelerometer “Sensor_Parameter” field

Table 8 lists the parameters of the accelerometer and the values of the “Sensor_Parameter” field.

Table 8. Accelerometer Sensor_Parameter field

Sensor_Parameter field	Parameter
0x00	Output data rate
0x01	Full scale
0x02	Acc_HPF
0x03	Offset_X
0x04	Offset_Y
0x05	Offset_Z
0x06 – 0xFF	Reserved for future use

2.4.5 Accelerometer output data rate

The "Parameter_Value" field for the output data rate setting is 1 byte in length. *Table 9* lists the supported output data rate for the accelerometer.

Table 9. Accelerometer output data rate and fields

“Parameter_Value” field for accelerometer ODR	Output data rate (Hz)
0x00	50
0x01	100
0x02	400
0x03	1000
0x04 – 0xFF	RFU

2.4.6 Accelerometer full scale

The "Parameter_Value" field for the full scale setting is 1 byte in length. *Table 10* lists the supported full scale for the accelerometer.

Table 10. Accelerometer full scale and fields

"Parameter_Value" field for accelerometer FS	Full scale (g)
0x00	±2 g
0x01	±4 g
0x03	±8 g
0x02, 0x04 – 0xFF	RFU

2.4.7 Accelerometer high-pass filter

The "Parameter_Value" field for the high-pass filter setting is 2 bytes in length as shown in *Figure 23*. *Table 11* indicates the possible cut-off frequencies.

Figure 23. Parameter_Value fields for accelerometer HPF setting

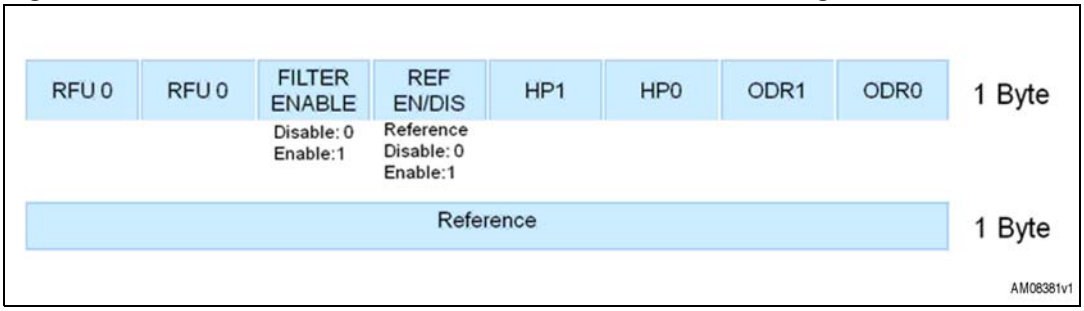


Table 11. Accelerometer high-pass filter setting

HP1	HP0	f_t[HZ] ODR = 00 data rate = 50 Hz	f_t[HZ] ODR = 01 data rate = 100 Hz	f_t[HZ] ODR = 10 data rate = 400 Hz	f_t[HZ] ODR = 11 Data rate = 1000 Hz
0	0	1	2	8	20
0	1	0.5	1	4	10
1	0	0.25	0.5	2	5
1	1	0.125	0.25	1	2.5

For further details please refer to the LSM303DLH datasheet.

2.4.8 Accelerometer offset

The "Parameter_Value" field for the offset (X, Y or Z axis) setting is 2 bytes in length and expressed in milli-g (thousandth of gravitational force) as signed short (16-bit), with the most significant byte first.

2.4.9 Magnetometer "Sensor_Parameter" field

Table 12 lists the values and parameters of the magnetometer "Sensor_Parameter" field.

Table 12. Magnetometer Sensor_Parameter field

Sensor_Parameter field	Parameter
0x00	Output data rate
0x01	Full scale
0x02	Operating mode
0x03	Offset_X
0x04	Offset_Y
0x05	Offset_Z
0x06 – 0xFF	RFU

2.4.10 Magnetometer output data rate

The "Parameter_Value" field for the output data rate setting is 1 byte in length. *Table 13* lists the values and the supported output data rate for the magnetometer.

Table 13. Magnetometer output data rate field

“Parameter_Value” field for magnetometer ODR	Output data rate (Hz)
0x00	0.75
0x01	1.5
0x02	3
0x03	7.5
0x04	15
0x05	30
0x06	75
0x07 – 0xFF	RFU

2.4.11 Magnetometer full scale

The "Parameter_Value" field for the full scale setting is 1 byte in length. *Table 14* lists the values and the supported full scale for the magnetometer.

Table 14. Magnetometer full scale field

“Parameter_Value” field for magnetometer FS	Full scale (Gauss)
0x01	±1.3
0x02	±1.9
0x03	±2.5