ISSDK Design Document for Host Interface

PRODUCT:	ISSDK
PRODUCT VERSION:	1.1
DESCRIPTION:	ISSDK Design Document for Host Interface
RELEASE DATE:	Dec 9th 2016

freescale™

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

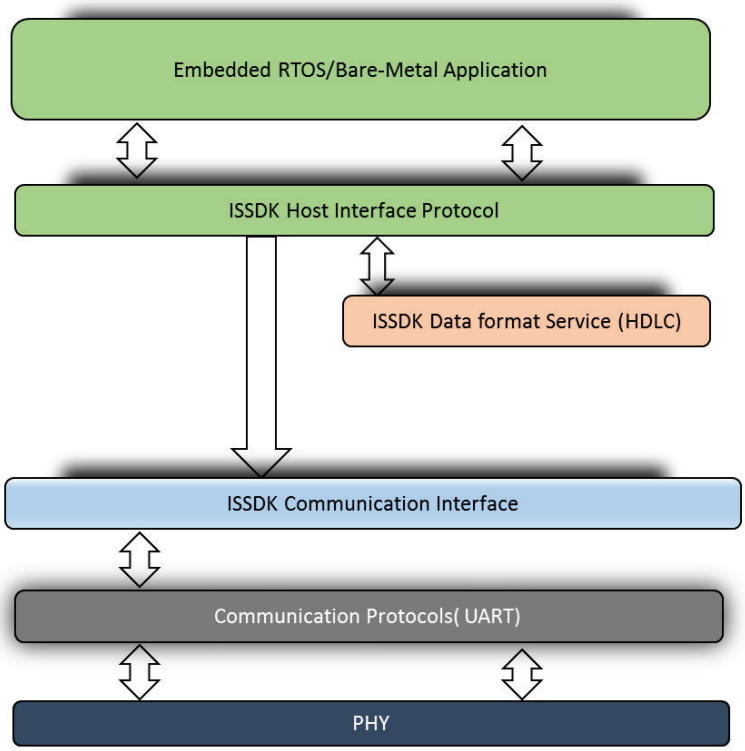
Rev. 09
01/2014

Table of Contents

1 Scope and Overview

This document summarizes the requirements and describes the design for a software service called the Host Interface. The Host Interface provides communication services between an ISSDK based embedded application and any host device over a Serial/UART interface.

The following figure shows the “layer cake” for the Host Interface implementation. At the lowest level, communication is supported by Kinetis SDK 2.0 UART drivers and the CMSIS Driver API interface (ISSDK Communication Interface and below). The ISSDK Host Interface Protocol uses the ISSDK Data format Service to provide HDLC style byte framing and character stuffing. The Host Interface provides functional APIs to the ISSDK Embedded application to access this service.



Architecture of ISSDK Host Interface

2 Host Interface Protocol

The ISSDK Host Interface Protocol provides three basic interface types of messages:

- Control Interface
    - Send/Response messages
- Isochronous Interface
    - Isochronous Command
- Discovery Interface
    - Device Info Send/Response

2.1 Control Interface

The Control interface provides for synchronous Send/Response message pairs. The Send/Response pair is typically used for control and status operations, but is extended to provide a form of Register Level Interface (RLI). The host should store the Sequence Identifier for each Send/Response message in order to error check missing responses. The embedded application processes each Send message and returns the Response.

Send/Response messages are defined for certain command types which include:

- Write App Data – provides data from the host to the embedded application (typically configuration)
- Read App Data – request data from the embedded application to the host (typically data or status)
- Write Register – used to set a sensor register value
- Read Register – used to read a sensor register value
- Customer – user defined extensions

2.1.1 Sequence Identifier

The Send/Response interface supports Sequence Identifiers to allow the Host to correlate Responses with the outstanding Commands. This implies that it is possible to write an ISSDK embedded application that could process Commands in non-FIFO order and respond. In this case, the Host is required to generate and insert unique and arbitrary Sequence Identifier into the Command messages and retain those values (i.e., the outstanding command context) until the response with the Sequence Identifier is returned.

Note: The Host may decide that it does not want to use Sequence Identifiers and may elect instead to only send the next command after the previous response has been returned. The advantage of using the Sequence Identifiers is to improve performance at the expense of managing the message context.

2.1.2 Send/Response Message Formats

Host Send Command Structure Detailed View

Field Name		7	6	5	4	3	2	1	0
Frame Tag	Name	Reserved	Interface Id		Command Type				
	Value		0x1		0x0 = Reserved 0x1 = Write App Data 0x2 = Read App Data 0x3 = Write Register Data 0x4 = Read Register Data 0x5 - 0x1F = Custom Commands				
Sequence Id									
Length									
PayLoad		Variable Length							

Host Response Command Structure Detailed View

Field Name		7	6	5	4	3	2	1	0
Frame Tag	Name	Send Command Status	Interface Id		Command Type				
	Value	0x1 = Command was executed successful 0x0 = command was not executed successful	0x1		0x0 = Reserved 0x1 = Write App Data 0x2 = Read App Data 0x3 = Write Register Data 0x4 = Read Register Data 0x5 - 0x1F = Custom Commands				
Sequence Id									
Length									
PayLoad		Variable Length							

Payload is shaded in gray in the examples provided.

2.1.3 Start Streaming Send/Response Example

This command is configuring the embedded application to start streaming data from the embedded application.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x21	Protocol	Snd/Rsp Interface Write App Data Cmd
Sequence Id	1	0x01	Host	Sequence Id assigned by host to correlate responses
Length	2	0x00 0x02	Protocol	Length of Payload in bytes
Operation Code	1	0x01	EmbApp	Start Command Value
Stream Id	1	0x01	EmbApp	Stream Identifier
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0xA1	Protocol	Cmd Successful Snd/Rsp Interface Write App Data Rsp
Sequence Id	1	0x01	Host	Sequence Id returned to Host to allow correlation of this response
Length	2	0x00 0x00	Protocol	Length of Payload in bytes
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Once streaming has started, the isochronous command format is used to transmit the data.

Those examples are provided.

2.1.4 Stop Streaming Send/Response Example

This command is configuring the embedded application to stop streaming data from the embedded application.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x21	Protocol	Snd/Rsp Interface Write App Data Cmd
Sequence Id	1	0x02	Host	Sequence Id assigned by host to correlate responses
Length	2	0x00 0x02	Protocol	Length of Payload in bytes
Operation Code	1	0x02	EmbApp	Stop Command Value
Stream Id	1	0x01	EmbApp	Stream Identifier
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0xA1	Protocol	Cmd Successful Snd/Rsp Interface Write App Data Rsp
Sequence Id	1	0x02	Host	Sequence Id returned to Host to allow correlation of this response
Length	2	0x00 0x00	Protocol	Length of Payload in bytes
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.1.5 Register Read Send/Response Example

This command is used to read a particular register on a particular device.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x24	Protocol	Snd/Rsp Interface Read Register Data Cmd
Sequence Id	1	0x03	Host	Sequence Id assigned by host to correlate responses
Length	2	0x00 0x03	Protocol	Length of Payload in bytes
Slave Address	1	0x1E	Host	I2C Slave Address (e.g. FXOS8700)
Register Offset	1	0x01	Host	Register Offset to start read (e.g. OUT_X_MSB)
Bytes to Read	1	0x06	Host	Number of Bytes to Read
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0xA4	Protocol	Cmd Successful/Snd/Rsp Interface Read Register Data Rsp
Sequence Id	1	0x03	Host	Sequence Id returned to Host to allow correlation of this response
Length	2	0x00 0x06	Protocol	Length of Payload in bytes
OUT_X_MSB	1	0xFF	EmbApp	X-Axis Accel MS byte
OUT_X_LSB	1	0xEC	EmbApp	X-Axis Accel LS byte
OUT_Y_MSB	1	0xFF	EmbApp	Y-Axis Accel MS byte
OUT_Y_LSB	1	0x3C	EmbApp	Y-Axis Accel LS byte
OUT_Z_MSB	1	0x03	EmbApp	Z-Axis Accel MS byte
OUT_Z_LSB	1	0xFC	EmbApp	Z-Axis Accel LS byte
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.1.6 Register Write Send/Response Example

This command is used to write to a particular register on a particular device.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x23	Protocol	Snd/Rsp Interface Write Register Data Cmd
Sequence Id	1	0x04	Host	Sequence Id assigned by host to correlate responses
Length	2	0x00 0x03	Protocol	Length of Payload in bytes
Slave Address	1	0x1E	Host	I2C Slave Address (e.g. FXOS8700)
Register Offset	1	0x2A	Host	Register Offset to start read (e.g. CTRL_REG1)
Value to Write	1	0x01	Host	Register Value (s) (e.g. Set ACTIVE bit)
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0xA3	Protocol	Cmd Successful Snd/Rsp Interface Write Register Data Rsp
Sequence Id	1	0x04	Host	Sequence Id returned to Host to allow correlation of this response
Length	2	0x00 0x00	Protocol	Length of Payload in bytes
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.2 Isochronous Interface

The Isochronous Interface is a light-weight interface meant for single, continuous and periodic data that does not have response data. There is no validation of the reliability of delivery and flow is not controlled. An Isochronous transaction can be initiated from either the host or the embedded application.

2.2.1 Streaming Message Format

Isochronous Command Structure Detailed View

Field Name		7	6	5	4	3	2	1	0
Frame Tag	Name	Reserved	Interface Id		Reserved				
	Value		0x2						
Length									
PayLoad		Variable Length							

Payload is shaded in gray in the examples provided.

2.2.2 Streaming Message Example

This is an example of streaming data that was started in an earlier example. Note that the format of the payload is determined by the embedded application implementation. The Host must adapt to the particular implementation in the embedded application.

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x40	Protocol	Streaming Interface
Length	2	0x00 0x17	Protocol	Length of Payload in bytes
Stream ID	1	0x01	EmbApp	The Stream ID for this Payload
Timestamp	4	0x00 0x00 0x00 0x00	EmbApp	Timestamp
Accel X-Axis	2	0xFF 0xFF	EmbApp	Accelerometer X-Axis Value
Accel Y-Axis	2	0xFF 0xFF	EmbApp	Accelerometer Y-Axis Value
Accel Z-Axis	2	0xFF 0xFF	EmbApp	Accelerometer Z-Axis Value
Mag X-Axis	2	0xFF 0xFF	EmbApp	Magnetometer X-Axis Value
Mag Y-Axis	2	0xFF 0xFF	EmbApp	Magnetometer Y-Axis Value
Mag Z-Axis	2	0xFF 0xFF	EmbApp	Magnetometer Z-Axis Value
Gyro X-Axis	2	0xFF 0xFF	EmbApp	Gyroscope X-Axis Value
Gyro Y-Axis	2	0xFF 0xFF	EmbApp	Gyroscope Y-Axis Value
Gyro Z-Axis	2	0xFF 0xFF	EmbApp	Gyroscope Z-Axis Value
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.3 Discovery Interface

The Discovery Interface allows the host to identify the software and hardware capabilities of a connected ISSDK embedded application running on a Freedom board, Sensor shield combination.

2.3.1 Device Info Send/Response Message Formats

Device Info Command Structure Detailed View

Field Name		7	6	5	4	3	2	1	0
Frame Tag	Name	Reserved	Interface Id		Reserved				
	Value		0x3						

Device Info Response Structure Detailed View

Field Name		7	6	5	4	3	2	1	0
Frame Tag	Name	Reserved	Interface ID		Reserved				
Value	Value		0x3						
ISSDK version (mandatory)		Major Version				Minor Version			
	Value								
Format	App Length	0-128							
	App String	e.g., "FXOS8700 Accelerometer Demo"							
	Board Length	0-128							
	Board String	e.g., "FRDM-K64F"							
	Shield Length	0-128							
	Shield String	e.g., "FRDM-STBC-AGM01"							

2.3.2 Device Info Example – Established Case

This example shows the Device Info command and response for the "Established" case. This means that the FRDM board has previously been programmed with an ISSDK compatible application and the hardware configuration has not changed since the last Device Info command.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Command
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Response
Host Protocol Version	1	0x10	Protocol	ISSDK Host Protocol Version (e.g. 1.0)
App String Length	1	0x1B	Protocol	Length of App String in bytes
App String		0x46 0x58 0x4F 0x53 0x38 0x37 0x30 0x30 0x20 0x41 0x63 0x63 0x65 0x6C 0x65 0x72 0x6F 0x6D 0x65 0x74 0x65 0x72 0x20 0x44 0x65 0x6D	EmbApp	ASCII String representing the Embedded Application (e.g. “FXOS8700 Accelerometer Demo”)

		0x6F		
Board String Length	1	0x09	Protocol	Length of Board String in bytes
Board String		0x46 0x52 0x44 0x4D 0x2D 0x4B 0x36 0x34 0x46	EmbApp	ASCII String representing the Embedded Application (e.g. “FRDM-K64F”)
Shield String Length	1	0x0F	Protocol	Length of Shield String in bytes
Shield String		0x46 0x52 0x44 0x4D 0x2D 0x53 0x54 0x42 0x43 0x2D 0x41 0x47 0x4D 0x30 0x31	EmbApp	ASCII String representing the Embedded Application (e.g. “FRDM-STBC-AGM01”)
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.3.3 Device Info Example – Swapped Case

This example shows the Device Info command and response for the “Swapped” case. This means the FRDM board is programmed with an ISSDK compatible application, but the sensor shield has changed since the last Device Info command.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Command
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Response
Host Protocol Version	1	0x10	Protocol	ISSDK Host Protocol Version (e.g. 1.0)
App String Length	1	0x1B	Protocol	Length of App String in bytes
App String		0x46 0x58 0x4F 0x53 0x38 0x37 0x30 0x30 0x20 0x41 0x63 0x63 0x65 0x6C 0x65 0x72 0x6F 0x6D 0x65 0x74 0x65 0x72 0x20 0x44 0x65 0x6D	EmbApp	ASCII String representing the Embedded Application (e.g. “FXOS8700 Accelerometer Demo”)

		0x6F		
Board String Length	1	0x09	Protocol	Length of Board String in bytes
Board String		0x46 0x52 0x44 0x4D 0x2D 0x4B 0x36 0x34 0x46	EmbApp	ASCII String representing the Embedded Application (e.g. “FRDM-K64F”)
Shield String Length	1	0x07	Protocol	Length of Shield String in bytes
Shield String		0x43 0x68 0x61 0x6E 0x67 0x65 0x64	EmbApp	ASCII String representing the Embedded Application (e.g. “Changed”)
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

2.3.4 Device Info Example – Undetectable Case

This example shows the Device Info command and response for the “Undetectable” case. Some sensor shield boards cannot be detected by the ISSDK compatible application. In that case they return this response.

Command:

Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Command
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

Response:

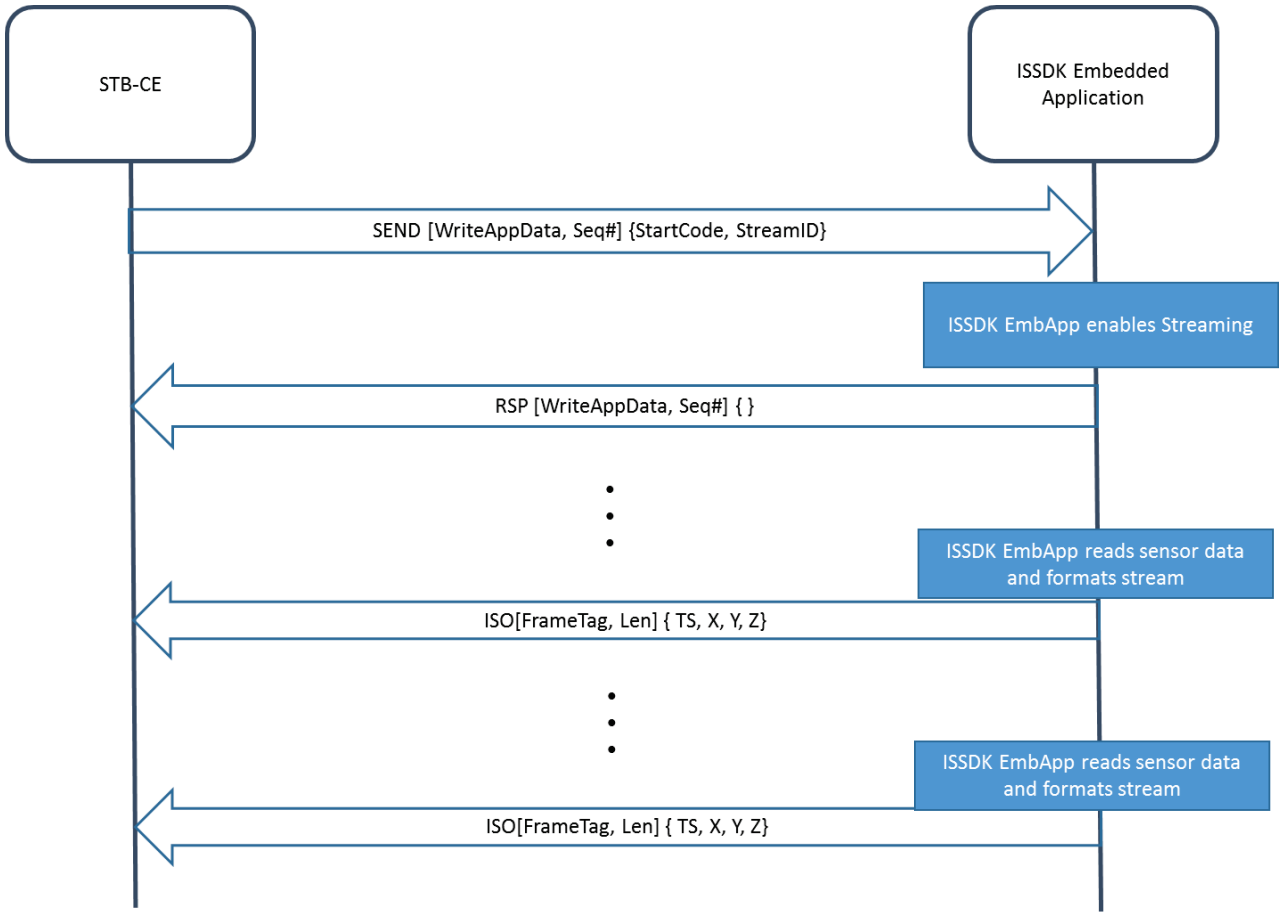
Field Name	Size (bytes)	Value	Value assigned by	Description
Start Character	1	0x7E	Protocol	Delimiter indicating start of packet
Frame Tag	1	0x60	Protocol	Discovery Interface Device Info Response
Host Protocol Version	1	0x10	Protocol	ISSDK Host Protocol Version (e.g. 1.0)
App String Length	1	0x1B	Protocol	Length of App String in bytes
App String		0x46 0x58 0x4F 0x53 0x38 0x37 0x30 0x30 0x20 0x41 0x63 0x63 0x65 0x6C 0x65 0x72 0x6F 0x6D 0x65 0x74 0x65 0x72 0x20 0x44 0x65 0x6D	EmbApp	ASCII String representing the Embedded Application (e.g. “FXOS8700 Accelerometer Demo”)

		0x6F		
Board String Length	1	0x09	Protocol	Length of Board String in bytes
Board String		0x46 0x52 0x44 0x4D 0x2D 0x4B 0x36 0x34 0x46	EmbApp	ASCII String representing the Embedded Application (e.g. “FRDM-K64F”)
Shield String Length	1	0x0C	Protocol	Length of Shield String in bytes
Shield String		0x4E 0x6F 0x74 0x20 0x44 0x65 0x74 0x65 0x63 0x74 0x65 0x64	EmbApp	ASCII String representing the Embedded Application (e.g. “Not Detected”)
Stop Character	1	0x7E	Protocol	Delimiter indicating end of packet

3 Example Message Flows

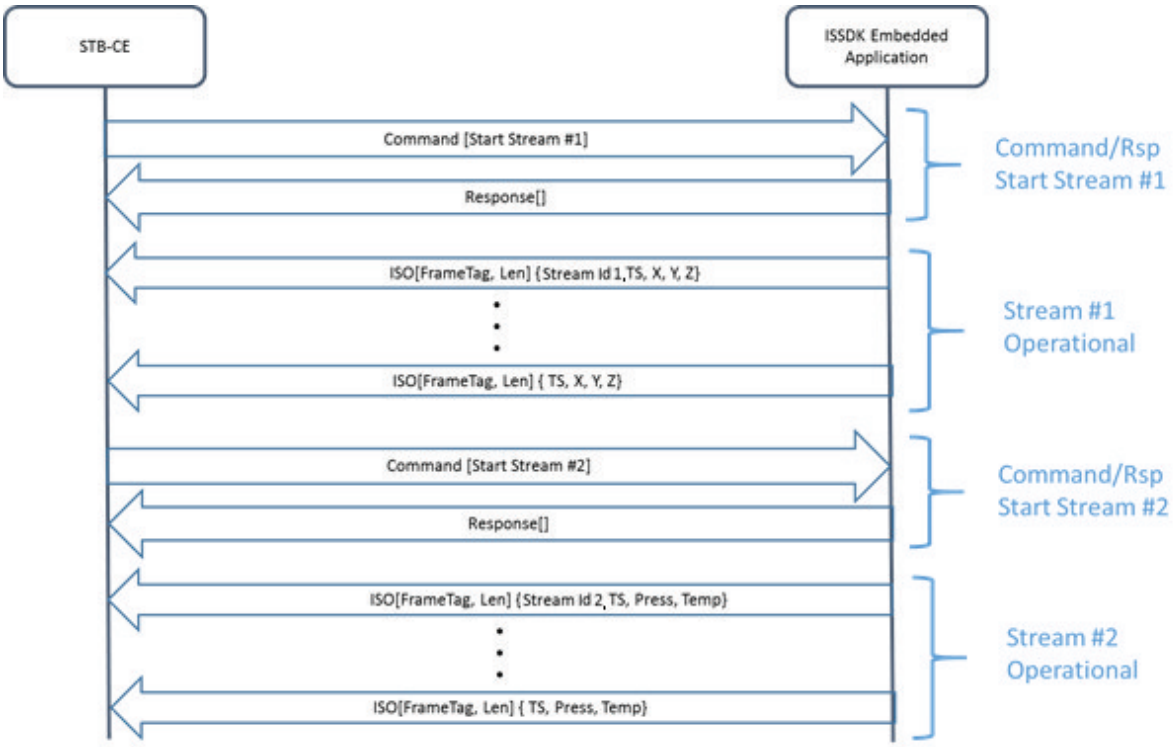
3.1 Start Streaming Data

In this message flow example, the Host sends a command to the EmbApp with a “Start Streaming” Operation Code and the StreamID of the stream to start. These values are assigned and interpreted by the EmbApp. They are arbitrary in terms of the Host Interface protocol. Upon receiving the command, the EmbApp enables streaming (Again, it is left to the EmbApp to determine how that is done) and responds to the host. At the desired epoch (typically a timer or interrupt from the sensor), the EmbApp reads the sensor data, formats it, and sends it in a Streaming message to the Host.



3.2 Start Multiple Streams

In this scenario, the Host starts two different streams (shown here as sequential streams). Each stream is controlled by unique Command messages, each with its own Operation Code and Stream Identifiers. The payload format for each stream is defined by the EmbApp and is interpreted by the Host.

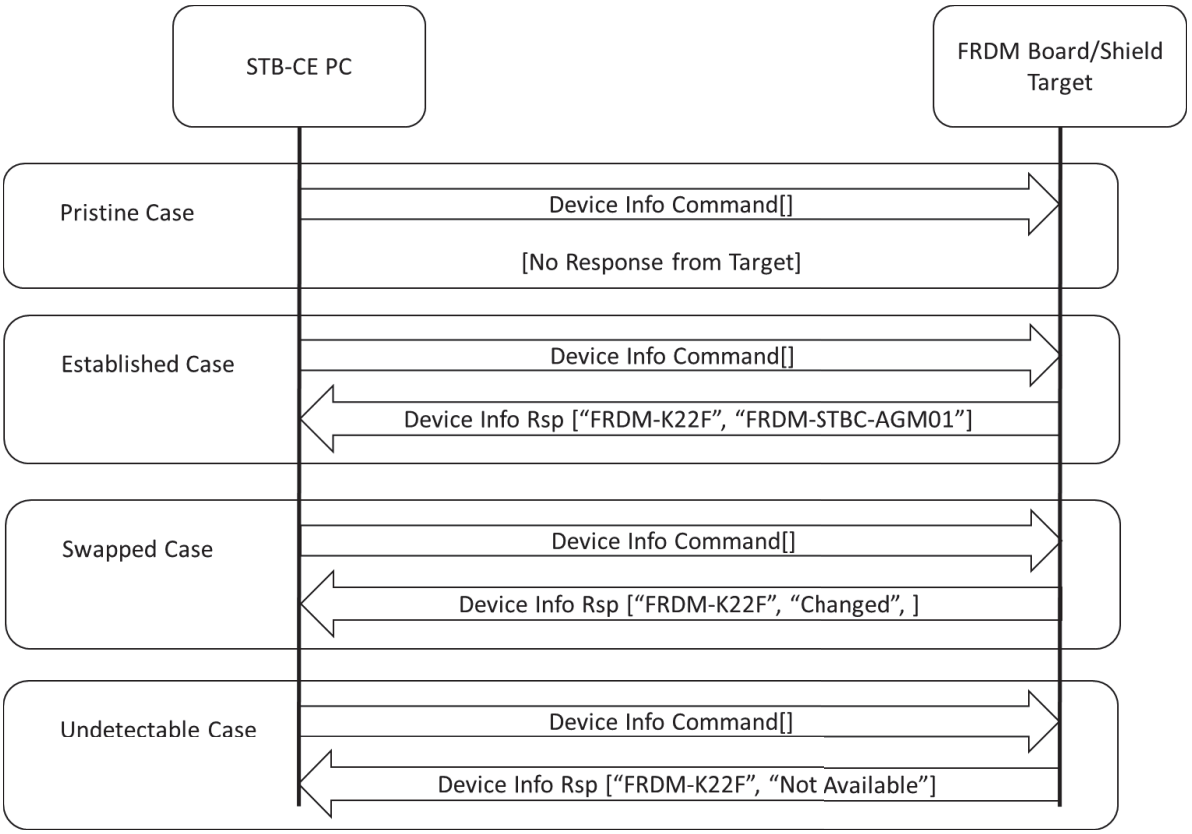


3.3 Discovery Interface Scenarios

There are four scenarios defined for Auto-Discovery:

1. Pristine Kit (new hardware; never connected to STB-CE)
   Response: No Response to Device Info
   Action: Query User for Board/Shield and Application to run
2. Established Kit (previously connected to STB-CE)
   Response: Responds with FRDM Board and Shield Info
   Action: Run Previous Application
3. Swapped Kit (new shield applied to Established FRDM board)
   Response: FRDM Board is recognized, but Shield indicates “Changed”
   Action: Query User for Board/Shield and Application to run
4. Undetectable Kit (analog or no WHOAMI shield)
   Response: FRDM Board is recognized, but Shield indicates “Not Available”
   Action: Query User for Board/Shield and Application to run

The following diagram shows the expected responses for each of these cases.



4 Host Interface API

The Host Interface API supported for the ISSDK applications is quite simple. It includes the following API function calls.

- Host_Initialize – Initializes the Host Interface for a specified communications channel and data formatting service (e.g. HDLC framing).
- Host_Configure – Configures the Host Interface.
- Host_Send – Sends a message/payload to the Host.
- Host_Receive – Receives a message/payload from the Host.

5 REVISION SHEET

Date	Owner/Reviewer	Description
10/21/2016	David Munsinger	Initial version.
10/26/2016	David Munsinger	Updated based on review comments from Sahil Choudhary (email
10/31/2016	David Munsinger	Reviewed Version for Release.
11/14/2016	Shubhadip Paul	Updated with post Review comments.
11/16/2016	David Munsinger	Added example for Device Info commands.