Kneron Inc

Document Name: **Host Interface Message Protocol**

Host Interface Message Protocol

Kneron Inc

Engineering Design Document

Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 Scope 3

2 Reference 3

3 Acronyms, Abbreviations, Definitions 3

4 System Architecture 4

4.1 Image Processing Data Flow 4

5 Message Syntax 5

5.1 Message Type 5

5.1.1 Test Commands (Memory and Control) 5

5.1.2 System Commands 6

5.1.3 System Command Responses 6

5.1.4 Application Specific Commands 7

5.2 Message Length 10

5.3 Message Body 10

5.4 Message Protocol Implementation 10

6 Message Description 11

6.1 Generic Message 11

6.1.1 Generic ACK Response Message to commands 11

6.2 Test Messages 12

6.2.1 Test Message Memory Read 12

6.2.2 Test Message Memory Write 13

6.2.3 Test Message Acknowledge/No-acknowledge 14

6.2.4 Test Message Memory Clear 15

6.2.5 Test Message Echo to Host 16

6.2.6 Test Message File Memory Write 17

6.2.7 Test Message FLASH Memory Write 18

6.3 System Messages 19

6.3.1 Reset 19

6.3.2 Report System Status 21

6.3.3 Update Firmware 22

6.3.3.1 Data Transfer Handshake 23

6.3.3.2 Binary Data Transfer Process 23

6.3.4 Update Model 24

6.4 FID Messages 25

6.4.1 Set FID Mode 25

6.4.2 Verify FID User 27

6.4.3 New FID User 28

6.4.4 Edit FID User 29

6.5 Simple FID Messages 31

6.5.1 Start SFID 32

6.5.2 New SFID User 33

6.5.3 Add New SFID User to DB 34

6.5.4 Delete SFID User DB 35

6.5.5 Edit SFID User DB 36
6.5.6 Send SFID Image 37
6.5.6.1 Image Data Transfer 38
6.5.6.2 Data Transfer Handshake 39
6.5.6.3 Data Transfer Format 39
6.5.7 Start LW3D FID 40
6.5.8 Send LW3D Image 41
6.6 Dynamic Model Execution Messages 43
6.6.1 Start DME 44
6.6.2 DME Configure 45
6.6.3 Send DME Image 46
6.6.3.1 DME Image Processing 47

1 Introduction

1.1 Purpose

The purpose of this document is to define the high-level host message protocol that will manage the communication between a Kneron devices and a host controller via a two-way message channel. It is intended to be layered on top of a low-level communication protocol for a specific physical interface which in turn carries out the actual message exchanges, such as a UART, SPI or some other future interfaces.

1.2 Scope

The protocol defined in this document shall covers all host communications between Kneron devices and the connected host controller. It shall apply to all applications, including testing and normal application processing. However, this document does not apply to the dedicated SPI interface used for chip testing or FLASH access purposes.

2 Reference

Kneron KL520 Design Specification, Feb. 2019
Host and Companion Communication Protocol, Kneron, Feb. 20, 2019

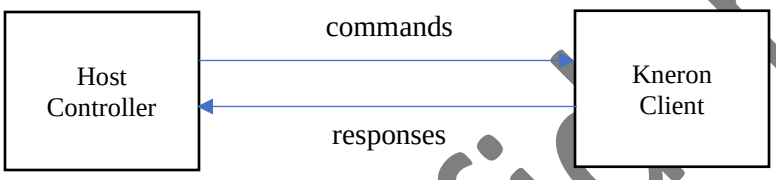
3 Acronyms, Abbreviations, Definitions

DME – Dynamic Model Execution
FID – Face Identification
HIMP – Host Interface Message Protocol
LW3D – Light Weight 3-Dimesions
NIR – Near InfraRed
RGB – Red Green and Blue (full color)
SFID – Simple Face Identification
SPI – Serial Peripheral Interface
UART – Universal Asynchronous Receiver-Transmitter

4 System Architecture

The physical message channel could be UART, SPI or some other future interface to be specified. The message protocol is structure from the point of view of a host controller. The commands are initiated from a host and transmitted to a Kneron device which will perform certain tasks and generate a reply. The message channel needs to be two way where each side may generate a message and transmit it.

Some physical interface such as SPI may be built on a master-slave model and in such cases the host controller shall be the bus master and support the receiving of messages from Kneron client via either an interrupt mechanism or through polling such that response message could be read on a timely fashion. The details of such message transactions shall be specified by interface-specific low-level message protocols in separate documents.



For every command issued, there shall be at least one response. In cases when the response will only be generated in a significant time later, a generic acknowledge shall be used to avoid the host from timing out and take recovery measures. Suggested response message timeout threshold is 100mS.

A command may generate multiple responses or asynchronous responses maybe generated as consequence of image processing or external events. Therefore, it is possible a response message maybe pending when a new command is being issued by the host. It is the host's responsibility to receive and process all responses from the Kneron client.

In addition, Kneron devices may support multiple host connections at the same time for different purposes. For example, on the KL520, a host connection to a device through a UART may exist to manage applications or debug operations but another host connection via SPI to a companion device may be used in addition for continuous image processing.

4.1 Image Processing Data Flow

The image processing flow envisioned by the message protocol consists of control message exchanges between the host controller and the Kneron client devices. It sets up the image processing models and the interfaces to be used for transmitting images to the Kneron devices.

The images themselves is not normally sent over this host interface except when using USB since the bandwidth required would normally be many times greater than the message exchanges that are expected over the host interface. For initial deployment, the images maybe sent into Kneron via either USB host or separate MIPI interfaces.

5 Message Syntax

type	length	message body

Each message is composed of a message type, length and the message body.

5.1 Message Type

A two-byte value is used to denote message type. Each valid message type is a command from the host and requires a corresponding response.

Valid commands are listed below in three categories:

5.1.1 Test Commands (Memory and Control)

Command	Message Type
CMD_MEM_READ	0x0001
CMD_MEM_WRITE	0x0002
CMD_ACK/NACK	0x0004
CMD_MEM_CLR	0x0006
CMD_TEST_ECHO	0x0008
CMD_FILE_WRITE	0x0009
CMD_FLASH_MEM_WRITE	0x000A

Test commands except ACK/NACK maybe disabled by eFuse so it will not be available in regular production units. For engineering development and early prototypes, these commands maybe available to support field debug needs.

5.1.2 System Commands

Command	Message Type
CMD_RESET	0x0020
CMD_SYSTEM_STATUS	0x0021
CMD_UPDATE_FW	0x0022
CMD_UPDATE_MODEL	0x0023

5.1.3 System Command Responses

The response to commands is specific to each command. The convention of a response message is to use the same Message Type as in the command received. This servers as a simple acknowledge to the reception of the command in most cases. If the most significant bit of the Message Type is also set, then a message body is also present and must be processed by the host.

For example, the command CMD_MEM_CLR, 0x0003 is issued, the expected response after the task is completed is a message of type 0x0003 with no message body. However, if the message type is 0x8003, then a message body is also present and needs to be processed. This could be used to indicate error status if errors were encountered.

The responses to operational commands that contain message bodies are listed below:

Response	Message Type
RESP_RESET	0x8020
RESP_SYSTEM_STATUS	0x8021
RESP_UPDATE_FW	0x8022
RESP_UPDATE_MODEL	0x8023

5.1.4 Application Specific Commands

Additional commands that are specific to application starts with 0x0100. They are grouped according to the specific applications. The Select Application command must be executed first before the corresponding application specific commands would be allowed.

Face Identification (FID)

These commands are used for Face Identification application and when activated by the Set FID Mode command will place the Kneron device into FID mode until the mode is exited, again via the Set FID Mode command.

The FID mode is latching, so once entered cannot be changed by either software or system resets, nor power cycles. While in this mode, some of the Operational Commands described above might be disabled to avoid conflicts.

Command	Message Type
CMD_SET_FID_MODE	0x0100
CMD_VERIFY_FID_USER	0x0101
CMD_NEW_FID_USER	0x0102
CMD_EDIT_FID_USER	0x0103

The responses to the FID commands that contain message bodies are listed below:

Response	Message Type
RESP _SET_FID_MODE	0x8100
RESP _VERIFY_FID_USER	0x8101
RESP _NEW_FID_USER	0x8102
RESP _EDIT_FID_USER	0x8103

Simple Face Identification (SFID)

These commands are used for Simple Face Identification application and are intended to support both the default 2-D face identification with a single RGB camera and the more advanced Light Weight 3D (LW3D) configuration with an additional NIR camera. For each configuration, there are two operating modes defined to support the use cases required for the application.

(1) Default mode, continuous FD/FR/DB operation. Performs FD-FR operation to extract a usable feature map from the incoming image frame and proceeds to 1:n DB comparison.
(2) DB operation, continuous FD-FR operation till a usable feature map is available from the incoming image stream. Once a feature map is found, it will be placed into a storage slot (max = 5) and wait for further instructions whose list is shown below. The current FD-FR operation can be terminated at any time by another SFID command.

- Perform another FD-FR.
- Register current feature maps. (up to 5)
- Delete all DB.

Command	Message Type
CMD_SFID_START	0x0108
CMD_SFID_NEW_USER	0x0109
CMD_SFID_ADD_DB	0x010A
CMD_SFID_DELETE_DB	0x010B
CMD_SFID_EDIT_DB	0x010F
CMD_SFID_SEND_IMAGE	0x010C
CMD_SFID_LW3D_START	0x010D
CMD_SFID_LW3D_IMAGE	0x010E

The responses to the SFID commands are listed below:

Response	Message Type
RESP_SFID_START	0x8108
RESP_SFID_NEW_USER	0x0109 / 0x8109
RESP_SFID_ADD_DB	0x810A
RESP_SFID_DELETE_DB	0x810B
RESP_SFID_EDIT_DB	0x810F
RESP_SFID_SEND_IMAGE	0x810C
RESP_SFID_LW3D_START	0x810D
RESP_SFID_LW3D_IMAGE	0x810E

Dynamic Model Execution (DME)

These commands are used for custom-designed inference applications that are downloaded from the host controller and executed under host control. There are two operating modes defined to support the download and execution of an application.

(1) Model and configuration download mode. Downloads the custom model and associated setup data, or the inference setup for image processing.
(2) Inference mode. Downloads an image for processing based on the inference setup previously downloaded.

Command	Message Type
CMD_DME_START	0x0118
CMD_DME_CONFIG	0x0119
CMD_DME_SEND_IMAGE	0x011A

The responses to the SFID commands that contain message bodies are listed below:

Response	Message Type
RESP_DME_START	0x8118
RESP_DME_CONFIG	0x8119
RESP_DME_SEND_IMAGE	0x811A

5.2 Message Length

The two-bytes length value contains the number of bytes (payload) the message body contains.

For example, a message body that is 1024 bytes, then the length field will be 0x04 and 0x00 followed by 1024 more bytes for the message body in a Little-Endian byte array.

For a message type that carries no payload, such as in simple acknowledgement, then the message length will be zero.

The maximum message body size supported by the protocol is thus 0xFFFF or 65535 bytes. But due to the actual message packet or buffer size limitation, actual message size maybe much smaller. To transfer messages larger than the packet or buffer limitation, message segmentation may need to be implemented at higher software level.

5.3 Message Body

The message body of commands is specific to each command and will be described in the next section.

The message body of responses is also specific to each command and will be described in sections pertinent to each command.

5.4 Message Protocol Implementation

Although the full set of Host Interface Message is defined in this document but actual devices may choose to only partially implement a subset of the messages defined. It is prudent to check the specific product design and implementation to determine which message is supported by the device.

Certain message types, such as those in the test command group, may be controlled by an eFuse and if so programmed, will be rejected for security reasons.

6 Message Description

For each command, there exists a simple ACK response with no message body. Such responses contain the same message type as in the commands with zero message length.

6.1 Generic Message

6.1.1 Generic ACK Response Message to commands

Name	RESP_ACK
u16 type	same as in command
u16 length	0
Body {	
}	

The simple ACK response is intended to support multi-thread non-blocking environments that may allow the same command to be issued multiple times before the previous one is completed. In such cases, the full responses will then be returned later as they are completed after sending out the ACK responses first.

Since the resource demand and workload involved in executing the command is not necessarily identical each time a command is issued, there should be no assumption about the time it takes to complete a command. Thus, the returned responses are not guaranteed to be in the same order. Hence in such cases, the command and response parameter list shall include a unique sequence or tag number to allow the correlating the responses with their specific command issuance.

Current existing commands and their normal responses are described in the following sections.

6.2 Test Messages

6.2.1 Test Message Memory Read

Name	CMD_MEM_READ
u16 type	0x0001
u16 length	8
Body {	
u32	starting address
u32	number of bytes to read
}	

Response Message to Memory Read

Name	RESP_MEM_READ
u16 type	0x8001
u16 length	8+n
Body {	
u32	Memory read result 0 – success else – error code
u32	number of bytes read
u8 * n	n bytes memory data
}	

In case of read error, the number of bytes read in the response will not equal to the command. The actual test module is responsible for performing the read operation. It is assumed the module has knowledge of the memory layout and would perform the read access correctly, i.e. perform appropriate byte, half-word, word access and use correct address alignment. It is also assumed the memory will terminate on the first memory address that fails and the error code would indicate the reason for failure. The error code encoding is defined by the test module.

6.2.2 Test Message Memory Write

Name	CMD_MEM_WRITE
u16 type	0x0002
u16 length	8+n
Body {	
u32	starting address
u32	number of bytes to write
u8 * n	n bytes memory data
}	

Response Message to Memory Write

Name	RESP_MEM_WRITE
u16 type	0x8002
u16 length	8
Body {	
u32	Memory write result 0 – success else – error code
u32	number of bytes written
}	

In case of write error, the number of bytes written in the response will not equal to the command. The actual test module is responsible for performing the write operation. It is assumed the module has knowledge of the memory layout and would perform the write access correctly, i.e. perform appropriate byte, half-word, word access and use correct address alignment. It is also assumed the memory will terminate on the first memory address that fails and the error code would indicate the reason for failure. The error code encoding is defined by the test module.

6.2.3 Test Message Acknowledge/No-acknowledge

Name	CMD_ACK/NACK
u16 type	0x0004
u16 length	8
Body {	
u32	command action 0 – ACK else - NACK
u32	source id
}	

The acknowledge command/response is used as a general acknowledge to the receiver instead of formatting a command-specific response as otherwise should be the case. This is done to accommodate the large number of test and demo scripts already written and used actively in the development environment.

It is also used as a quick handshake message to initiate a follow-on transaction outside the normal command/response mechanism.

The command action field indicates whether the ACK/NACK command is used as ACK (action = 0) or NACK (action = stop/abort). When a NACK command action is indicated, the field also encodes the reason why the original command is aborted.

The source id field is used to indicate the context from which the ACK/NACT originates.

Additional NACK codes are defined below:

250 – memory allocation failure
255 – incorrect image processing mode configuration

6.2.4 Test Message Memory Clear

Name	CMD_MEM_CLR
u16 type	0x0006
u16 length	8
Body {	
u32	starting address
u32	number of bytes to clear
}	

Response Message to Memory Clear

Name	RESP_MEM_CLR
u16 type	0x8006
u16 length	8
Body {	
u32	Memory read result 0 – success else – error code
u32	number of bytes cleared
}	

In case of memory clear error, the number of bytes cleared in the response will not equal to the command. The actual test module is responsible for performing the clear operation. It is assumed the module has knowledge of the memory layout and would perform the write access correctly, i.e. perform appropriate byte, half-word, word access and use correct address alignment. It is also assumed the memory will terminate on the first memory address that fails and the error code would indicate the reason for failure. The error code encoding is defined by the test module.

6.2.5 Test Message Echo to Host

Name	CMD_TEST_ECHO
u16 type	0x0008
u16 length	8+n
Body {	
u32	reserved
u32	number of bytes to echo
u8 * n	n bytes test data
}	

Response Message to Echo to Host

Name	RESP_TEST_ECHO
u16 type	0x8008
u16 length	8+n
Body {	
u32	Echo test result 0 – success else – error code
u32	number of bytes echoed
U8 * n	n bytes test data
}	

The behavior of the Echo to Host command is actually defined by the test module. The Host Interface Message Protocol is written in a way to support either character I/O or line I/O cases. The test mode setting may need to be implemented in the low-level device drivers, for example, as part of the UART or SPI drivers.

The HIMP facilitates the entry into the Echo Test mode with the Echo to Host command but exiting from the echo test mode is undefined. Again, it may need to be implemented in the low-level device drivers. A response message with success and zero byte echoed, i.e. the message length value is 4, could be used as confirmation of the exit from the echo test mode.

6.2.6 Test Message File Memory Write

Name	CMD_FILE_WRITE
u16 type	0x0009
u16 length	8
Body {	
u32	starting address
u32	number of bytes to write
}	

Response Message to File Memory Write

Name	RESP_FILE_WRITE
u16 type	0x8009
u16 length	8
Body {	
u32	Memory write result 0 – success else – error code
u32	number of bytes written
}	

The File Write command is intended to transfer large amount of data, such as the images, into the Kneron device. The memory starting address must be 32-bit aligned. The effective starting address thus will have the lower 2 bits masked off from the starting_address field of the command.

In case of write error, the number of bytes written in the response will not equal to the command. The actual test module is responsible for performing the write operation. It is assumed the module has knowledge of the memory layout and would perform the write access correctly, i.e. perform appropriate byte, half-word, word access and use correct address alignment. It is also assumed the memory will terminate on the first memory address that fails and the error code would indicate the reason for failure. The error code encoding is defined by the test module.

6.2.7 Test Message FLASH Memory Write

Name	CMD_FLASH_MEM_WRITE
u16 type	0x000A
u16 length	8+n
Body {	
u32	starting FLASH address
u32	number of bytes to write
u8 * n	n bytes FLASH data
}	

Response Message to FLASH Memory Write

Name	RESP_FLASH_MEM_WRITE
u16 type	0x800A
u16 length	8
Body {	
u32	Memory write result 0 – success else – error code
u32	number of bytes written
}	

The FLASH MEM Write command is used to store data in the FLASH device. The data to be written should all be contained in the same 4KB sector. The maximum number of bytes to be written is also limited to 4KB. This command will cause the original data in the FLASH to be written over. If the FLASH write process was disrupted in any way for whatever reason, an irreversible data corruption may result.

All parameters of the FLASH Memory Write command must be aligned to 32-bit address boundary. Therefore, the number of bytes to write must also be multiples of 4-bytes.

Currently defined error codes for this command are:

1 – FLASH address out of range
2 – data verification failed

If repeated FLASH MEM Write to the same sector persistently returns error code 2, **data verification failed**, it is an indication that the sector might be bad, perhaps due to excessive wear. The sector then should be marked as bad and avoided in further data storage use.

6.3 System Messages

6.3.1 Reset

Name	CMD_RESET
u16 type	0x0020
u16 length	8
Body {	
u32	reset mode control 0 – no operation 1 – reset message protocol 3 – enter suspended mode 4 – return active operation 255 – reset system 256 – system shut down (RTC) 0x1000xxxx – reset debug output Level 0x2000xxxx – reset & switch active boot partition
u32	Confirm reset mode control Value must be 1's complement of the reset mode control
}	

Response to Reset

Name	RESP_RESET
u16 type	0x8020
u16 length	8
Body {	
u32	error code 0 – no error, command mode is 0 1 – undefined reset control mode 2 – invalid confirmation of mode
u32	Reserved
}	

A response to the Reset command will be transmitted first with the system reset and restart operation after a pre-defined delay duration. Reset mode 256, system shut down, is one exception in that it will not proceed to automatic restart after performing the shutdown operation.

Reset mode 3 and 4 (suspended mode) are only valid for UART host interface; it is undefined for USB host interface. For UART hosts, reset mode 3 will cause the system to enter a low power status. Any subsequent command will automatically return the system to active mode. No reset mode 4 command is currently needed.

For reset mode 255 (reset system), the message protocol will be afterwards in the initial power up state.

For reset mode 256 (system shut down), there would be no restart of the system. An external start/restart mechanism is required to relaunch the system.

If reset mode is 0x1000xxxx, then the debug output level is configured by the 'xxxx' value. When xxxx = 0x0000, it means no debug log output is enabled.

If reset mode is 0x2000xxxx, then the KL520 will reset and restart with a different firmware partition than the one it currently uses. To switch scpu boot image, set bit 0 to 1 and for ncpu, set bit 1 to 1. For bits that are zero, no switching will occur.

When the KL520 restarts, it would be using the new boot image as specified in this case. Repeating the same command a second time will cause the KL520 to return to the original image since currently there are only two images defined for each component inside KL520.

6.3.2 Report System Status

Name	CMD_SYSTEM_STATUS
u16 type	0x0021
u16 length	0
Body {	
}	

The Report System Status is intended for use after a system reset in order to find out information about the device and its resources status. It is also used to determine when the firmware on the device has completed the reset process and is ready to receive additional commands.

The report status command could also be used at any time after reset to determine the device status.

Response to Report System Status

Name	RESP_SYSTEM_STATUS
u16 type	0x8021
u16 length	12
Body {	
u32	device id
u32	firmware id
u16	system status bit 0 – suspended operation
u16	application status application id
}	

Currently, only four reporting fields are defined: device id, firmware id, system status information and the active application id.

The firmware id is the firmware revision number consists of a major, a minor, a patch, and an edit number.

The application ids currently defined are as below:

1 – LW3D
2 – SFID
3 - DME

It is expected additional fields will be defined and included in this message in the future. As a result, the message length value may increase in later revisions of this document with additional fields defined.

6.3.3 Update Firmware

Name	CMD_UPDATE_FW
u16 type	0x0022
u16 length	8
Body {	
u32	firmware module id 0 – no operation 1 – scpu 2 – ncpu
u32	reserved
}	

The Update Firmware command will start the process to download, verify and program the code into KL520's internal flash memory for subsequent use.

The size of the firmware module is pre-defined and could not be changed. The host file size to be downloaded must exactly match the pre-defined size or the download process will fail.

The initial response to Update Firmware will be an ACK/NACK command and is described in more details below.

After the completion of the module download and subsequent programming into flash, the response message for the command would be returned to the host. There would be no automatic reload and launch of the newly downloaded firmware code.

Response to Update Firmware

Name	RSP_UPDATE_FW
u16 type	0x8022
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	firmware module id
}	

If there is no error, then the module id field will contain the confirmation of the module id of the module programmed into flash. If the command specified module id = 0, no data transfer would take place and instead of the acknowledge message, the full response message will be returned with no error code.

Additional error code(s) are defined below:

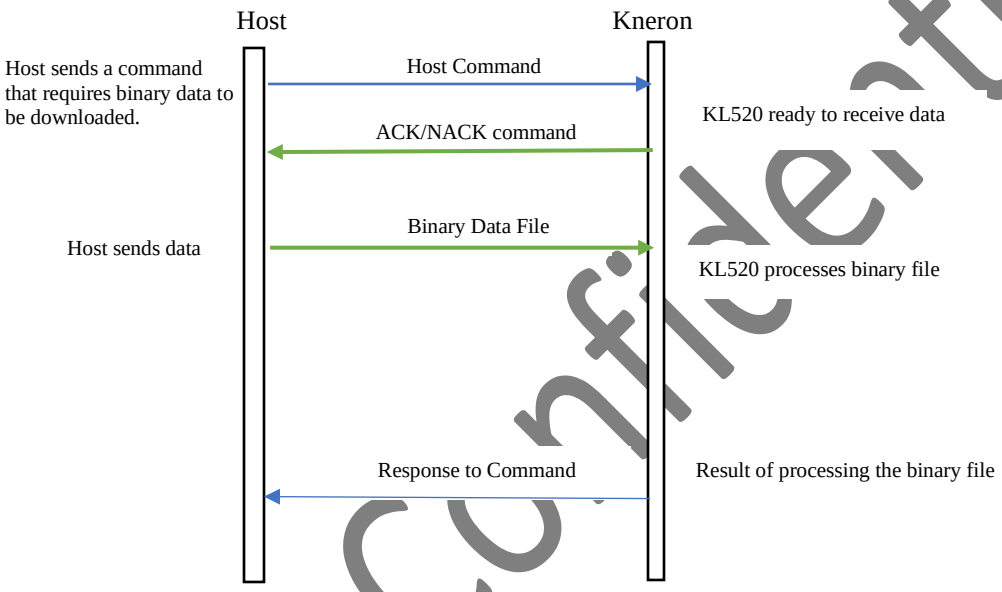
251 – module download failed authentication.
252 – flash programming failed.
253 – module download failed.

6.3.3.1 Data Transfer Handshake

The ACK/NACK message is used to acknowledge the start of image data transfer by the KL520. If the message indicates no error, then the host is expected to proceed with the data download as soon as possible. If there is error code returned, then the image data transfer will not be carried out and the host should abort the data transfer and process the error code accordingly.

6.3.3.2 Binary Data Transfer Process

The image transfer process is shown in more detail below:



6.3.4 Update Model

Name	CMD_UPDATE_MODEL
u16 type	0x0023
u16 length	8
Body {	
u32	0 – no operation else – model id
u32	size of model
}	

The Update Model command will initiate the process to download, verify and program the model into KL520's internal flash memory for subsequent use.

The size of the model file to be downloaded must exactly match the actual size to be transferred from the host or the download process will fail.

The model download process uses the same binary file download mechanism as that for the Update Firmware command and the initial response to Update Model will be an ACK/NACK command which is described in more details in the Update Firmware section above.

After the completion of the model download and subsequent programming into flash, the response message for the command would be returned to the host.

Response to Update Model

Name	RESP_UPDATE_MODEL
u16 type	0x8023
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	model id
}	

If there is no error, then the model id field will confirm the id of the model programmed into flash.

Additional error code(s) are defined below:

251 – model update failed authentication.
252 – flash programming failed.
253 – model update failed.

6.4 FID Messages

FID messages are used by the Application Manager to support the Face ID application. The message protocol is not yet finalized and is subject to change in the future.

6.4.1 Set FID Mode

Name	CMD_SET_FID_MODE
u16 type	0x0100
u16 length	8 or 12
Body {	
u16	Select new FID mode 0 – de-activate FID 1 – normal FID operation 2 – edit existing user 3 – add new user 128 – enter configuration mode 255 - change user password
u16	verification code 0 – use user password 1 – use unit verification code
u32	password or code
u32	[new user password] or [configuration parameters]: 1 – self-test 10 – factory reset 20 – low-power (standby mode)
}	

The Set FID Mode command will place the Kneron device into a specific operating mode. There are four active FID modes defined. The Kneron device can be in one of the modes but its functionality is restricted to those specified in the specific mode. To switch to a different functionality that exists in a different mode, the Set FID Mode command must be executed first.

Normal FID

This is the default FID mode when FID is activated by entering into any of the three active FID modes. While in the Normal FID mode, the Kneron device will perform image capture and Face Identification when so commanded.

Edit FID User

The Edit FID User mode provides access to the current registered user database stored within the Kneron device. This mode is timer controlled so after a predetermined inactivity duration, the device will automatically revert back to the Normal FID Mode.

Add New User

The Add New User mode allows the image capturing and registration of a new user in the user database within the Kneron device. This mode is also timer controlled and will revert back to the Normal FID Mode after certain inactivity duration.

Enter Configuration Mode

The Configuration mode is used to perform system functions, such as self-test, factory reset, power-down mode, change communication parameters or other system functions to be defined in the future. Factory

reset will not erase the user image database. This mode is also timer controlled and will revert back to the Normal FID Mode after certain inactivity duration.

Deactivate FID

The deactivate option will place the Kneron device in a non-FID status and the default system operations will apply. FaceID functions are not available while in this state.

Unit Verification Code is a unique code related to the serial number of the Kneron device programmed during device initialization and is unalterable afterwards. This code is not user selectable nor configurable.

There is only a single user password defined currently. No multiple user support is currently planned.

Response to Set FID Mode

Name	RESP_SET_FID_MODE
u16 type	0x8100
u16 length	20
Body {	
u32	Response code 0 – no error else – error code
u16	new FID mode
u16	reserved
u8*12	unit information
}	

The actual error code is encoded by the image processing module. The unit information field and its size are tentative at this time. This field may only be relevant when in the Configuration Mode. The unit information field right now is intended to deliver the serial number of the device.

6.4.2 Verify FID User

Name	CMD_VERIFY_FID_USER
u16 type	0x0101
u16 length	8
Body {	
u16	subcommand 0x0002 – check against all users 0x2912 – check against single user
u16	reserved
u16	User ID
u16	user_image_index
}	

The Verify User command will setup an image path specified by the app_id field in the command. Once an image is placed into the image buffer then a 1:N Face ID result is generated. If a single user mode is selected, the comparison is performed for the specified user ID-image_index. If all-users mode is selected, the comparison is performed over all users in the database in which case the user ID and/or user_image_index is ignored.

This command is only valid in the Normal FID or Add New User mode.

Response to Verify FID User

Name	RESP_VERIFY_FID_USER
u16 type	0x8101
u16 length	8
Body {	
u32	Response code 0 – no error 10 – image capture timeout 20 – no image match
u16	user ID
u16	reserved
}	

The actual error code is encoded by the image processing module. If no uer ID match is found, either due to no match or image capture timeout, an error code is returned. In such cases, the user ID value shall be ignored. This response also serves as an abnormal FID notification.

6.4.3 New FID User

Name	CMD_NEW__FID_USER
u16 type	0x0102
u16 length	8
Body {	
u16	sub command 0x20 – capture image 0x22 – register image
u16	image_index
u16	reserved
u16	reserved
}	

The capture image command will setup an image path specified by the app_id field in the command. Once an image is streamed into the Kneron device, it will be kept in the buffer memory.

The image_index field is used to assign a specific storage slot for the new user. The valid index value is 1 through 5.

The register image command will store all the image data into the device FID database and assigns a user ID. Valid User ID value will be from 1 to 15 and is assigned by the device.

The register image command will erase all current images in the image buffer.

Response to New FID User

Name	RESP_NEW_FID_USER
u16 type	0x8102
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u16	user_ID
u16	user_image_index
}	

The actual error code is encoded by the image processing module. If no storage is available, an error code is returned instead.

Additional error codes may specify timeout or reaching retry-limit without successful ID.

User_ID is only valid when the subcommand issued was 0x22, register image.

6.4.4 Edit FID User

Name	CMD_EDIT__FID_USER
u16 type	0x0103
u16 length	8
Body {	
u16	sub command 0x49 – list all users 0x44 – delete single user 0x105 – delete all users 0x22 – export user images
u16	user ID
u16	user_image_index
u16	reserved
}	

This command is only valid in the Edit FID User mode.

The user ID field is only used if sub command 0x44, delete single user, is issued. Otherwise it is not relevant and would be ignored.

The user_image_index field is only used for sub command 0x22, export user data. This is needed to limit the amount of data in the response message.

Response to Edit FID User

Name	RESP_EDIT_FID_USER
u16 type	0x8103
u16 length	8+n
Body {	
u32	Response code 0 – no error else – error code
u16	subcommand repeated 0x22 – export user data 0x44 – delete single user 0x49 – list all users 0x105 – delete all users
u16	for 0x22 – user ID for 0x44 – new number of users for 0x49 – number of users for 0x105 – new number of users
u8*n	for 0x22, n = 1028 for 0x49, n = number of users*2
}	

The list all users sub command will generate a list of currently registered FID users of the device. The number of users that can be supported is 15 at this point. Each user ID is a u16. It is assumed the list would fit into a single response message.

Number of users in the response reflects the new number of users after the command has completed.

The export user sub command will export the feature map data for one user-image combination with an object size of 1028 bytes, with a two-byte image_index followed by 1024 bytes of feature map.

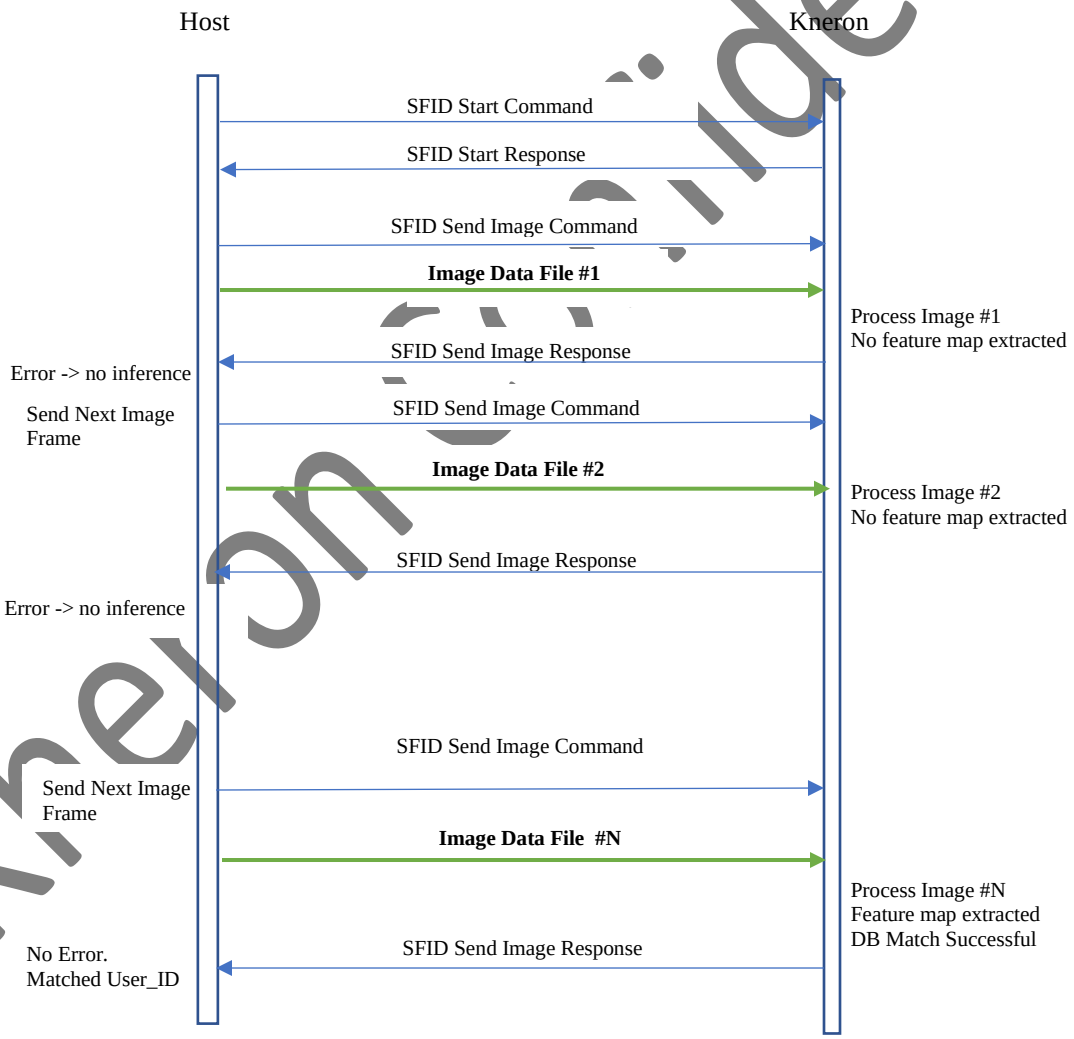
6.5 Simple FID Messages

Simple FID messages are used by the Application Manager to support the Simple FID application which is intended to demonstrate the capability of the KL520 device. The image path utilizes the same path as the command/response interface. The message process flow is depicted in the below diagram.

While SFID Start command is in effect, the Sent SFID Image command and responses are repeated until interrupted by another SFID command.

The Add SFID User command is similar except that the once a final response message is generated, which could include a number of fatal errors when adding the new user, would stop the Add SFID User process and would require another Add SFID User command to re-start the process again.

The host uses SFID Send Image command to inform KL520 the upload of a new image file to the KL520 which will start off the image processing steps. The image transfer process is described in more details in the SFID Send Image Command section later.



6.5.1 Start SFID

Name	CMD_SFID_START
u16 type	0x0108
u16 length	16
Body {	
float32	FR threshold
u32	reserved
u16	image width
u16	image height
u32	image format
}	

The Start SFID command will start the FD-FR process until a valid feature map is obtained from the image stream. A user DB 1:N comparison will then be performed. The operation is continuous so once the DB comparison is finished, another round of FD-FR will be initiated automatically.

The FR threshold is the value used to determine the minimum acceptable score for a facial recognition match. This value is expressed in single precision float format (IEEE 754) and maybe adjusted according to the confidence level required for a specific application (business critical needs). If a value of zero is used, then a built-in default threshold will be used instead. Normal range of the FR threshold is 0.0-1.0.

Image width is number of pixels horizontally (x-axis) and height vertically (y-axis). Image format is a 32-bit value specified by Kneron.

Response to Start SFID

Name	RESP_SFID_START
u16 type	0x8108
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	image_size
}	

If there is no error, then the image_size field will contain the size in bytes of the expected image size.

Additional error code(s) are defined below:

100 – SFID application not loaded.
255 – Image format not supported.

6.5.2 New SFID User

Name	CMD_SFID_NEW_USER
u16 type	0x0109
u16 length	8
Body {	
u16	User_ID
u16	reserved
u16	Image_index (1-5)
u16	reserved
}	

The New SFID User command will start the FD-FR process until a valid feature map is obtained from the image stream. No DB comparison will be performed. The operation is one shot so once a final response is generated or a fatal error encountered, the Application Manager will stop and wait for further instruction.

The New SFID User command is expected to be followed by multiple SFID Send Image commands to deliver an image stream until a usable feature map is identified. (See Section 6.6)

The New SFID User command should not be exercised until after a valid Start SFID command is executed. This is to ensure the image size is established correctly between the host and the KL520 device.

The User_ID is intended to associate the current image capturing session with a user identity. Valid value for the User_ID is 1-20. Changes to the User_ID value when building up the image database before adding the user to the database will result in errors.

Valid Image_index value is 1-5. A New User session must start with Image_index value of 1. Otherwise an error will be returned.

If the User_ID already exists in the image database, an error will be returned. For the User_ID to be used again, it must be deleted from the database first.

If the User_ID is available, the feature map data is saved into a temporary buffer as specified by the command. Later when an Add New SFID User command is executed, it and all other data from this session will be stored in the FLASH or other suitable NVM.

Response to New SFID User

Name	RESP_SFID_NEW_USER
u16 type	0x0109
u16 length	0
Body {	
}	

This response serves as an acknowledgement to the host that the add new user process has commenced. The KL520 would then wait for the images to be uploaded from the host.

6.5.3 Add New SFID User to DB

Name	CMD_SFID_ADD_DB
u16 type	0x010A
u16 length	4
Body {	
u32	User_ID
}	

The Add New SFID User command will move the current feature maps in the temporary storage for the new user into NVM memory as specified by the User_ID. If the User_ID is not the same as the one used for the temporary storage, the command will result in error and will not be executed. The host could issue Delete User command first if the intent is to replace the user data in the database. Or start another New User session by issuing a New SFID User command with the intended User ID.

Response to Add New SFID User to DB

Name	RESP_SFID_ADD_DB
u16 type	0x810A
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	reserved
}	

If there is no error, then the feature maps accumulated is added to the database successfully.

Additional error code(s) are defined below:

255 – Image processing not configured.
258 – User ID already exists in database.
259 – User ID changed from the New User session.
261 – No feature maps from New User session could be found.
263 – DB add failure.

6.5.4 Delete SFID User DB

Name	CMD_SFID_DELETE_DB
u16 type	0x010B
u16 length	4
Body {	
u32	User_ID
}	

The Delete SFID User DB command will erase a single user's current feature maps in the database as specified by the User_ID value. If the User_ID value of zero is used, then all user data will be erased. This action is not reversable.

Valid User_ID value for this command is 0-20. A value of zero will delete the entire user DB.

The Delete SFID User DB command will not alter the temporary data stored in the current New User session.

Response to Delete SFID User DB

Name	RESP_SFID_DELETE_DB
u16 type	0x810B
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	reserved
}	

If there is no error code, then the user's feature maps are deleted from the database.

Additional error code(s) are defined below:

255 – Image processing not configured.
262 – User ID not found in DB.
263 – DB delete failure.

6.5.5 Edit SFID User DB

Name	CMD_SFID_EDIT_DB
u16 type	0x010F
u16 length	8 + [4]
Body {	
u32	DB action 1 – import 2 – export
u32	reserved
[u32]	if DB action = 1 size of DB file
}	

The Edit SFID User DB command will accept a new or export the existing DB image from/to the host. The DB action field indicates the operation to be performed on the DB. Currently, only export and import DB are defined. Additional action or authentication maybe defined in the future.

Response to Edit SFID User DB

Name	RESP_SFID_EDIT_DB
u16 type	0x810F
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	if no error and DB action = 2 size of DB file
}	

If there is no error code and the DB action in command is “export”, then the entire DB file is to be sent out via the interface immediately.

Additional error code(s) are defined below:

248 – DB action/format not supported.
249 – DB export failed.
251 – DB authentication error.
253 – DB import failed.
255 – DB access not configured.

6.5.6 Send SFID Image

Name	CMD_SFID_SEND_IMAGE
u16 type	0x010C
u16 length	8
Body {	
u32	image_file_size
u16	output enable mask
u16	reserved
}	

The Send SFID Image command will inform KL520 that the download of an image data file by the host is about to start. The image file is uploaded as raw binary data. The number of bytes specified by image_file_size must match the actual number of bytes transferred.

A Start SFID command must have been executed prior to issuing the Send SFID Image. Otherwise the image processing resources would not have been allocated inside KL520 and the Send SFID Image command will result in Image Processing Not Configured error (error code 255).

The output enable flag will generate FD/FR intermediate outputs if enabled. Bit 0 is used for FD results, bit 1 is for landmark data and bit 2 for FR feature map.

Response to Send SFID Image

Name	RESP_SFID_SEND_IMAGE
u16 type	0x810C
u16 length	12 + [8] + [20] + [2048]
Body {	
u32	Response code 0 – no error else – error code
u16	0 – normal mode else – new user mode
u16	normal mode – User ID new user mode – Image index
u16	output data flag bit 0 – FD result bit 1 – LM data bit 2 – FR feature map
u16	reserved
[u8 * 8]	If output data flag bit 0 = 1 optional FD result
[u8 * 20]	If output data flag bit 1 = 1 optional LM data
[u8 * 2048]	If output data flag bit 2 = 1 optional FR data
}	

If there is no error code, then a feature map is extracted from the image. Either a User ID, in the normal SFID mode, or an Image_index, in the new SFID user mode, will be included in the UserID/Image_index field.

The output data flag is used to indicate if additional FD/FR results are available. Bit 0 of the data flag indicates whether the 8-byte FD result is appended to the response message. Bit 1 indicates whether the 10-byte LM result is appended to the response message. Bit 2, if set, indicates a 2KB feature map data is appended to the response message.

The error code(s) for the response message are defined below:

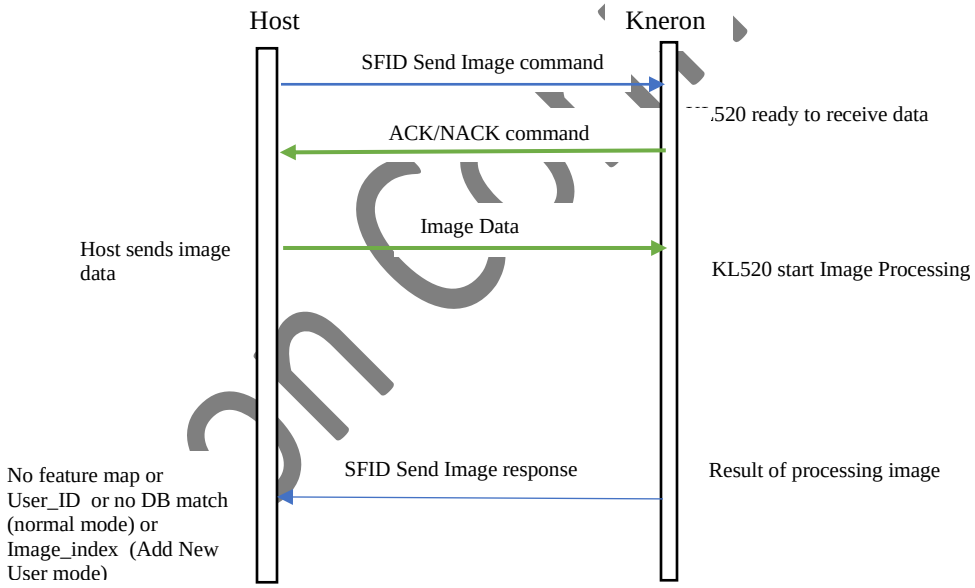
10 – No match in User DB.
255 – Image processing not configured.
256 – No successful inference (FD-FR)

And while in the Add User session, the following error codes maybe encountered:

255 – Image processing not configured.
257 – Invalid User ID or image index specified.
258 – User ID already exists.
259 – User ID changed during this New User session.

6.5.6.1 Image Data Transfer

The image transfer process is shown in more detail below:



6.5.6.2 Data Transfer Handshake

The ACK/NACK message is used to acknowledge the start of image data transfer by the KL520. If the message indicates no error, then the host is expected to proceed with the data download as soon as possible. If there is error code returned, then the image data transfer will not be carried out and the host should abort the data transfer and process the error code accordingly.

6.5.6.3 Data Transfer Format

The raw binary image data is transferred to KL520 as one data block via USB or in multiple blocks via UART until the entire image file is transferred. No other command or action may take place until the transfer ends.

The data block is transmitted by the host as binary data without any header or descriptor. The data block is not formatted as messages and thus is not subject to the requirements defined in Section 5 of this document.

For normal image transfers, the size of data transferred shall be the exact size in bytes as specified in the Start SFID response message.

6.5.7 Start LW3D FID

Name	CMD_SFID_LW3D_START
u16 type	0x010D
u16 length	24
Body {	
float32	primary image FR threshold
float32	secondary image FR threshold
u16	primary image width
u16	primary image height
u32	primary image format
u16	secondary image width
u16	secondary image height
u32	secondary image format
}	

The Start Light Weight 3D (LW3D) command is similar to the Start SFID command and will start the FD-FR process until valid feature maps are obtained from the image stream. A user DB 1:N comparison will then be performed. The operation is continuous so once the DB comparison is finished, another round of FD-FR will be initiated automatically.

The FR thresholds are the values used to determine the minimum acceptable score for a facial recognition match. This value is expressed in single precision float format (IEEE 754) and maybe adjusted according to the confidence level required for a specific application (business critical needs). If a value of zero is used, then a built-in default threshold will be used instead. Normal range of the FR threshold is 0.0-1.0.

The image width, height and format are specified individually for each primary and secondary camera. When one camera is RGB and the other is NIR, the RGB camera shall be the primary and the NIR shall be the secondary camera.

Response to Start LW3D

Name	RESP_SFID_LW3D_START
u16 type	0x810D
u16 length	12
Body {	
u32	Response code 0 – no error else – error code
u32	primary image size
u32	secondary image size
}	

If there is no error, then the Image size fields will contain the size in bytes of the expected image size.

Additional error code(s) are defined below:

100 – LW3D application not loaded.
255 – Image format not supported.

6.5.8 Send LW3D Image

Name	CMD_SFID_LW3D_IMAGE
u16 type	0x010E
u16 length	12
Body {	
u32	primary image_file_size
u32	secondary image_file_size
u16	output enable mask
u16	reserved
}	

The Send SFID Image command will inform KL520 that the download of image data files by the host is about to start. The image file is downloaded as raw binary data and there are two cycles of image file download, the first for Primary (or RGB) image and the second for the secondary (NIR) image. The number of bytes specified by image_file_size must match the actual number of bytes to be transferred.

A Start LW3D command must have been executed prior to issuing the Send LW3D Image. Otherwise the image processing resources would not have been allocated inside KL520 and the Send LW3D Image command will result in Image Processing Not Configured error (error code 255).

The output enable flag will generate FD/FR intermediate outputs if enabled. Bit 0 is used for Primary FD results, bit 1 for Primary landmark data, bit 2 for Primary feature map, bit 3 is reserved, bit 4 for Secondary FD results, bit 5 for Secondary landmark data, bit 6 for Secondary FR feature map, bit 7 is reserved and bit 8 for the final, combined Liveness result.

Response to Send LW3D Image

Name	RESP_SFID_SEND_IMAGE
u16 type	0x810E
u16 length	12 + [8] + [20] + [2048] [8] + [20] + [2048] + [4]
Body {	
u32	Response code 0 – no error else – error code
u16	0 – normal mode else – new user mode
u16	normal mode – User ID new user mode – Image index
u16	output data flag bit 0 – Primary FD result bit 1 – Primary LM data bit 2 – Primary feature map bit 4 – Secondary FD result bit 5 – Secondary LM data bit 6 – Secondary feature map bit 8 – Final Liveness result
u16	Reserved
[u8 * 8]	If output data flag bit 0 = 1 optional primary FD result
[u8 * 20]	If output data flag bit 1 = 1 optional primary LM data

[u8 * 2048]	If output data flag bit 2 = 1 optional primary FR result
[u8 * 8]	If output data flag bit 4 = 1 optional secondary FD result
[u8 * 20]	If output data flag bit 5 = 1 optional secondary LM data
[u8 * 2048]	If output data flag bit 6 = 1 optional secondary FR result
[u16*2]	If output data flag bit 8 = 1 optional two u16 fields first u16 Final Liveness result 0 – Liveness Test Failed 1 – Liveness Test Passed second u16 reserved
}	

If there is no error code, then a feature map is extracted from the image. Either a User ID, in the normal SFID mode, or an Image_index, in the new SFID user mode, will be included in the UserID/Image_index field.

The output data flag is used to indicate if additional FD/FR results are available. Bit 0 & 4 of the data flag indicates whether the 8-byte FD result is appended to the response message. Bit 1 & 5 indicates whether the 10-byte LM result is appended to the response message. Bit 2 & 6 indicates whether the 2KB feature map data is appended to the response message. And bit 8, if set, indicates the final Liveness result is appended to the response message.

The error code(s) for the response message are defined below:

10 – No match in User DB.
255 – Image processing not configured.
256 – No successful inference (FD-FR)

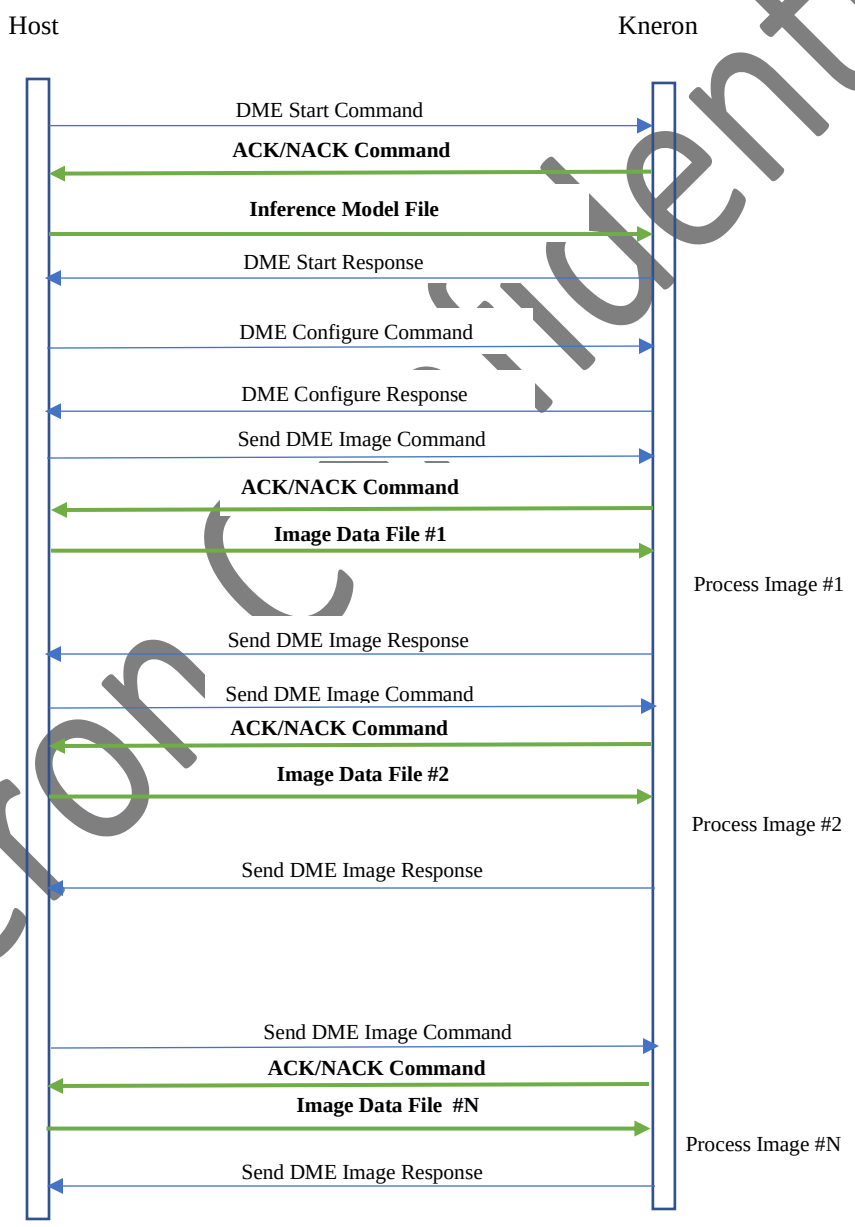
And while in the Add User session, the following error codes maybe encountered:

255 – Image processing not configured.
257 – Invalid User ID or image index specified.
258 – User ID already exists.
259 – User ID changed during this New User session.

6.6 Dynamic Model Execution Messages

Dynamic Model Execution messages are used for custom-designed inference applications that are downloaded from the host controller and executed under host control. There are two operating modes defined to support the download and execution of an application.

(1) Model and configuration download mode. Downloads the custom model and associated setup data, or the inference setup for image processing.
(2) Inference mode. Downloads an image for processing based on the inference setup previously downloaded.



6.6.1 Start DME

Name	CMD_DME_START
u16 type	0x0118
u16 length	4 + [n]
Body {	
u32	size of inference model
u8 [n]	firmware setup data
}	

The Start DME command will start the process to set up and download a host controller-supplied inference model for image processing.

The firmware setup data is used to configure the KL520 internal resources to manage and exercise the various models that embedded within the inference model file being downloaded as part of the same command. Due to the internal data alignment needs, the size of inference model and the setup data must be multiple of 4 bytes.

The initial response to Start DME is the Start DME Response message. If there is no error code, then the inference model download shall proceed thereafter as soon as possible. The inference model file is downloaded as raw binary data. The number of bytes specified by size_of_inference_file must match the actual number of bytes to be transferred. The details of the binary file download is similar to other command processing that also requires binary file downloads. See the message flow diagram in Section 6.6.5.1 as a useful reference.

After the completion of the model download, the response message for the command would be returned to the host.

Response to Start DME

Name	RESP_DME_START
u16 type	0x8118
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	inference_model_size
}	

If there is no error, then the inference_model_size field will confirm the size in bytes of the received model.

Additional error code(s) are defined below:

253 – data download failed.

6.6.2 DME Configure

Name	CMD_DME_CONFIG
u16 type	0x0119
u16 length	4+[n]
Body {	
u32	setup configuration
u8 [n]	inference setup data
}	

The DME configure command is used to deliver the inference setup data necessary to control the processing of the image data. Due to the internal data alignment needs, the size of inference setup data must be multiple of 4 bytes.

Setup configuration parameter specifies how the setup data is to be utilized.

Response to DME Configure

Name	RESP_DME_CONFIG
u16 type	0x8119
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	reserved
}	

If there is no error code, then the inference set up data are processed successfully.

Additional error code(s) are defined below:

255 – DME processing not configured.

6.6.3 Send DME Image

Name	CMD_DME_SEND_IMAGE
u16 type	0x011A
u16 length	8
Body {	
u32	image file size
u32	reserved
}	

The Send DME Image command will inform KL520 that the upload of an image data file by the host is about to start. The image file is uploaded as raw binary data. The number of bytes specified by image file size field must match the actual number of bytes transferred.

A Start DME and DME Configure command must have been executed prior to issuing the Send DME Image. Otherwise the image processing configuration would not have been set up correctly inside KL520 and the Send DME Image command will result in Image Processing Not Configured error (error code 255).

Response to Send DME Image

Name	RESP_DME_SEND_IMAGE
u16 type	0x811A
u16 length	8
Body {	
u32	Response code 0 – no error else – error code
u32	size of inference results
}	

If there is no error code and inference results are available, the size of the inference results will be non-zero. The host controller shall then start the result data upload by issuing an ACK/NACK command. Upon receiving the ACK message by KL520, the transmission of KL520 result data will subsequently commence.

Additional error code(s) are defined below:

255 – Image processing not configured.

6.6.3.1 DME Image Processing

The result data transfer process is shown in more detail below: