ST25 fast transfer mode embedded library

Introduction

The ST25 fast transfer mode (ST25FTM) library is an embedded software library enabling fast data transfer between an NFC reader and a dynamic tag. The dynamic tag features a low latency memory that can be read and written by both an NFC reader and an MCU connected to the dynamic tag.

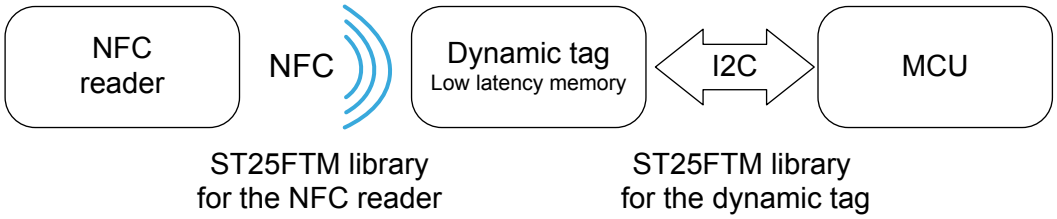
The ST25FTM library leverage on this low latency memory to exchange messages from the NFC reader to the MCU and the other way around.

The ST25FTM protocol has been designed to manage such data transfer, using as less meta-data as possible, while enabling error detection and recovery.

The ST25FTM library is an embedded implementation of the ST25FTM protocol that can be used on both an NFC reader and the MCU controlling the dynamic tag.

The ST25FTM library is part of the ST25 NFC embedded library package (STSW-ST25R-LIB). The STSW-ST25R-LIB embedded software provides middlewares and their associated examples that can be reused when developing an application with ST25R products.

Figure 1. Fast transfer mode overview



1 ST25FTM library examples

The ST25FTM library comes with examples running on the following hardware:

- X-NUCLEO-NFC06A1 featuring the ST25R3916 NFC reader.
- X-NUCLEO-NFC05A1 featuring the ST25R3911 NFC reader.
- X-NUCLEO-NFC04A1 featuring the ST25DV-I2C dynamic tag.
- X-NUCLEO-NFC03A1 featuring the ST25R95 NFC reader.

The NFC reader example starts polling for a ST25DV-I2C, and when detected, it initiates the transfer by transmitting a large amount of data. The dynamic tag example receives this data, switching the blue LED on, and once the transfer has completed, the dynamic tag example sends the data back to the NFC reader, switching the green LED on. In case of error the dynamic tag example switch the yellow LED on.

The NFC reader may be removed for a short period without jeopardizing the current transfer. If the NFC reader is kept away from the tag for a longer period, the transfer is reset and the NFC reader re-start polling for a ST25DV-I2C.

2 ST25FTM protocol

The low latency memory involved in the fast transfer mode has a defined length which is usually smaller than the length of the data to be transferred. The ST25FTM protocol splits the data to be transferred into messages fitting into the low latency memory.

The protocol manages the message chaining process using the first byte of all frames. This byte is named the control byte and each of its bit provides an information about the transfer state (see below description).

The ST25FTM protocol also provides the option to check the data integrity after a certain number of messages have been sent. The bunch of messages on which integrity is checked is called a segment.

ST25FTM control byte:

- b7 **message type**: set to 0 for a control byte.
- b6 **packet length**: when this bit is set, the packet length field is present in the message. This field indicates the length of the current message in bytes (including the CRC Bytes). The packet length field is necessary when the length of the message is smaller than the low-latency memory size. The Packet length field, when present, is appended after the control byte.
- b5 **segment end**: a CRC is present in this packet. The CRC is calculated on all the data present between “segment start” and “segment end” messages. An Acknowledge message is expected from the receiver as the next message.
- b4 **segment start**: The current message is the first message of a segment. The CRC computation must restart form this packet.
- b3-b2 **packet position**: These bits are used to indicate if the transfer requires several messages, and if yes it defines the position of the packet:
  - 00: The transfer length fit in a single message.
  - 01: First packet of a transfer. In that case, the Total Length field is present, next after the control byte. The Total Length field indicates the total payload length for the transfer in bytes.
  - 10: Middle packet of the transfer.
  - 11: End packet, finishing the transfer.
- b1 **segment id**: This bit toggles each time a segment has been successfully acknowledged, to facilitate detection of a retransmitted segment.
- b0: This bit indicates if the packet belongs to a segment.

When the transmitter has emitted a message with the segment end bit set (and the CRC was appended), the receiver is expected to reply with an acknowledge message. This acknowledge message is identified by its very first byte which is called the Status Byte.

ST25FTM Status byte

- b7 **message type**: set to 1 for a Status Byte.
- b6-b2: reserved for future use.
- b1-b0 **status**: acknowledge response for the received Segment.
  - 00 segment OK: the data integrity has been successfully checked (CRC).
  - 01 segment Error: the data integrity has failed.

Upon the reception of a successful acknowledge message, the transmitter continues by transmitting the first packet of the next Segment, toggling the segment Id bit from the control byte.

In case of unsuccessful acknowledge, the transmitter restarts and sends the previous segment again.

3 ST25FTM library

The ST25FTM library provides an implementation of the protocol described in Section 2 ST25FTM protocol that can be run on an MCU. This The ST25FTM API can be called by an embedded application to transfer data using the fast transfer mode of a dynamic tag. The FTM transfer runs in a non-blocking way and progresses when the application runs the ST25FTM worker.

3.1 ST25FTM API

The ST25FTM library offers an API to the application to control the FTM transfers. This API is implemented in the st25ftm_protocol.c source file.

Table 1. ST25FTM_Init function

Function	void ST25FTM_Init(void)
Parameters	No parameters.
Returns	No return value.
Description	This function initializes both: • the ST25FTM library itself. • the device, by calling the Interface function.

Table 2. ST25FTM_SendCommand function

Function	void ST25FTM_SendCommand(uint8_t* data, uint32_t length, ST25FTM_Send_Ack_t ack, ftm_data_cb data_cb)
Parameters	data pointer: a pointer to the data buffer to be transmitted. length: number of bytes to be transmitted. ack: it enables hand-checks during the transmitter. • ST25FTM_SEND_WITHOUT_ACK: no acknowledge required. • ST25FTM_SEND_WITH_ACK: acknowledge is required. data_cb: optional callback function, it is called to request data to send (to be set to NULL if not used).
Returns	No return value.
Description	The function initializes a transfer to the peer device.

Table 3. ST25FTM_ReceiveCommand function

Function	void ST25FTM_ReceiveCommand(uint8_t* data, uint32_t *length, ftm_data_cb data_cb)
Parameters	data pointer: a pointer to the data buffer. It is used for data reception. length pointer: a pointer to a word defining the maximum number of bytes that can be received. This parameter is also used to return the number of bytes actually read. ack: it enables hand-checks during the transfer data_cb: optional callback function. It is called to write the received data (to be set to NULL if not used).
Returns	No return value.
Description	The function initializes a reception from the peer device.

Table 4. ST25FTM_Runner function

Function	void ST25FTM_Runner(void)
Parameters	No parameters.
Returns	No return value.
Description	This function is the runner for the ST25FTM library. It must be run regularly to let the library send or receive the data.

Table 5. ST25FTM_Status function

Function	ST25FTM_State_t ST25FTM_Status(void)
Parameters	No parameters.
Returns	Return the current state of the ST25FTM lib (ST25FTM_State_t enum) ST25FTM_IDLE: No communication on-going. ST25FTM_WRITE: A transmission is on-going. ST25FTM_READ: A reception is on-going.
Description	This function returns the current state of the ST25FTM library.

Table 6. ST25FTM_SetTxFrameMaxLength function

Function	void ST25FTM_SetTxFrameMaxLength(uint32_t len)
Parameters	len: set the maximum message length (in bytes).
Returns	No return value.
Description	This function sets the maximum message length for the transmission.

Table 7. ST25FTM_SetRxFrameMaxLength function

Function	void ST25FTM_SetRxFrameMaxLength(uint32_t len)
Parameters	Len: set the maximum message length (in bytes).
Returns	No return value.
Description	This function sets the maximum message length for the reception.

Table 8. ST25FTM_SetRxFrameMaxLength function

Function	uint32_t ST25FTM_GetTxFrameMaxLength(void)
Parameters	No parameter.
Returns	Return the current maximum message length (in bytes) for the transmission.
Description	This function returns the maximum message length for the transmission.

Table 9. ST25FTM_GetRxFrameMaxLength function

Function	uint32_t ST25FTM_GetRxFrameMaxLength(void)
Parameters	No parameter.
Returns	Return the current maximum message length (in bytes) for the reception.
Description	This function returns the maximum message length for the reception.

Table 10. ST25FTM_IsNewFrame function

Function	uint8_t ST25FTM_IsNewFrame(void)
Parameters	No parameter.
Returns	Return 1 the first time this function is called after a reception began, 0 if not.
Description	This function is used to let the application knows that a reception has started.

Table 11. ST25FTM_GetFieldState function

Function	ST25FTM_Field_State_t ST25FTM_GetFieldState(void)
Parameters	No parameter.
Returns	Return the current field state (ST25FTM_Field_State_t enum): ST25FTM_FIELD_OFF: the RF field is off. ST25FTM_FIELD_ON: the RF field is on.
Description	This function returns the current field state.

Table 12. ST25FTM_GetTransferProgress function

Function	uint32_t ST25FTM_GetTransferProgress(void)
Parameters	No parameter.
Returns	Return the current transfer progress in percentage from 0 to 100.
Description	This function computes and returns the progress of the current transfer.

Table 13. ST25FTM_GetAvailableDataLength function

Function	uint32_t ST25FTM_GetAvailableDataLength(void)
Parameters	No parameter.
Returns	Return the number of received bytes available (and validated).
Description	This function returns the number of received bytes.

Table 14. ST25FTM_GetReadBufferOffset function

Function	uint8_t ST25FTM_GetReadBufferOffset(uint8_t *dst, uint32_t length)
Parameters	dst: buffer to copy the data. length: number of bytes to be read.
Returns	Return the status: 0 if success, 1 if not.
Description	This function reads the specified number of bytes from the reception buffer.

Table 15. ST25FTM_ReadBuffer function

Function	uint8_t ST25FTM_ReadBuffer(length)
Parameters	length: number of bytes to be transmitted.
Returns	Return the offset of the next byte to read.
Description	This function gets the current offset in the command of the next byte, which is read with ST25FTM_ReadBuffer.

Table 16. ST25FTM_GetTotalLength function

Function	uint32_t ST25FTM_GetTotalLength(void)
Parameters	No parameter.
Returns	Return the total length of the current transfer (in bytes).
Description	This function returns the total length of the current transfer (in bytes), including the CRC.

Table 17. ST25FTM_GetRetryLength function

Function	uint32_t ST25FTM_GetRetryLength(void)
Parameters	No parameter.
Returns	Return the length of data that has been resent (in bytes).
Description	This function returns the length of data that has been resent (in bytes).

Table 18. ST25FTM_IsReceptionComplete function

Function	uint8_t ST25FTM_IsReceptionComplete(void)
Parameters	No parameter.
Returns	Return 1 if the reception has completed, 0 if not.
Description	This function returns the reception completion status.

Table 19. ST25FTM_IsTransmissionComplete function

Function	uint8_t ST25FTM_IsTransmissionComplete(void)
Parameters	No parameter.
Returns	Return 1 if the transmission has completed, 0 if not.
Description	This function returns the transmission completion status.

Table 20. ST25FTM_CheckError function

Function	uint8_t ST25FTM_CheckError(void)
Parameters	No parameter.
Returns	Return 1 if a fatal error occurred, 0 if not.
Description	This function returns the error status.

Table 21. ST25FTM_IsIdle function

Function	uint8_t ST25FTM_IsIdle(void)
Parameters	No parameter.
Returns	Return 1 if the ST25FTM library is idle, 0 if a transfer is running.
Description	This function returns the ST25FTM library idle status.

Table 22. ST25FTM_Reset function

Function	void ST25FTM_Reset(void)
Parameters	No parameter.
Returns	No return value.
Description	This function resets the ST25FTM library state.

Table 23. ST25FTM_SetTxSegmentMaxLength function

Function	uint8_t ST25FTM_SetTxSegmentMaxLength(uint32_t length)
Parameters	Length: number of bytes to be transmitted.
Returns	The maximum value is ST25FTM_SEGMENT_LEN.
Description	This function sets the length of a segment in bytes.

Table 24. ST25FTM_ResetTxSegmentMaxLength function

Function	uint8_t ST25FTM_ResetTxSegmentMaxLength(uint32_t length)
Parameters	Length: number of bytes to be transmitted.
Returns	The maximum value is ST25FTM_SEGMENT_LEN.
Description	This function resets the length of a segment in bytes. The maximum value is ST25FTM_SEGMENT_LEN.

Table 25. ST25FTM_GetTxSegmentMaxLength function

Function	uint8_t ST25FTM_GetTxSegmentMaxLength(uint32_t length)
Parameters	Length: number of bytes to be transmitted.
Returns	The maximum value is ST25FTM_SEGMENT_LEN.
Description	This function gets the length of a segment in bytes.

3.2 ST25FTM middleware interface file

The ST25FTM library is delivered as a STM32Cube middleware. This means that the library is made independent from the Hardware: MCU, NFC reader and dynamic tag. An interface header file defines the lowest layer API for the middleware, to be implemented for the targeted hardware.

A template for this Interface header file is available here:

`Middlewares/ST/ST25FTM/st25ftm_config_template.h.`

The Interface files for the following STMicroelectronics NFC readers and dynamic tag are available in the ST25FTM library software package:

- Projects/STM32L476RG-Nucleo/Applications/X-NUCLEO-NFC0xA1/FTM/Src/st25r_st25dv-i2c_ftm.c implementation is the same for all NFC readers:
  - X-NUCLEO-NFC06A1 featuring the ST25R3916 NFC reader
  - X-NUCLEO-NFC05A1 featuring the ST25R3911 NFC reader
  - X-NUCLEO-NFC03A1 featuring the ST25R95 NFC reader
- Projects/STM32L476RG-Nucleo/Applications/X-NUCLEO-NFC04A1/FTM/NFC/Targetst25dv-i2c_ftm.c for the X-NUCLEO-NFC04A1 featuring the ST25DV-I2C Dynamic Tag

The interface for the ST25FTM middleware is as follow:

Table 26. ST25FTM_DeviceInit function

Function	int ST25FTM_DeviceInit(void)
Parameters	No parameters.
Returns	Integer value: 0 for a success, any other value in case of failure.
Description	This function initializes the device for the Fast Transfer Mode.

Table 27. ST25FTM_SetDevice function

Function	void ST25FTM_SetDevice(rfalNfcvListenDevice *device)
Parameters	Device: a pointer to a rfalNfcvListenDevice structure.
Returns	No return value.
Description	This function sets the detected NFC tag (only for NFC readers).

Table 28. ST25FTM_CRC_Initialize function

Function	void ST25FTM_CRC_Initialize(void)
Parameters	No parameters.
Returns	No return value.
Description	This function initializes the CRC IP.

Table 29. ST25FTM_GetCrc function

Function	uint32_t ST25FTM_GetCrc(uint8_t *data, uint32_t length)
Parameters	Data: pointer to a buffer of uint8_t on which to compute the CRC. Length: the number of bytes in the data buffer.
Returns	The CRC value on 4 bytes.
Description	This function computes the CRC on the given data.

Table 30. ST25FTM_UpdateFieldStatus function

Function	void ST25FTM_UpdateFieldStatus(void)
Parameters	No parameters.
Returns	No return value.
Description	This function updates an internal variable with the current RF field value.

Table 31. ST25FTM_GetMessageOwner function

Function	ST25FTM_MessageOwner_t ST25FTM_GetMessageOwner(void)
Parameters	No parameters.
Returns	The owner of the message in the memory (ST25FTM_MessageOwner_t enum): ST25FTM_MESSAGE_EMPTY: the buffer is free. ST25FTM_MESSAGE_ME: busy with a message from this device. ST25FTM_MESSAGE_PEER: busy with a message from peer device. ST25FTM_MESSAGE_OWNER_ERROR: status unknown.
Description	This function detects the owner of the current message.

Table 32. ST25FTM_ReadMsg function

Function	ST25FTM_Msg_Status_t ST25FTM_ReadMsg(uint8_t *msg, uint32_t* msg_len)
Parameters	Msg: a buffer to be used to store the read message. Msg_len: a pointer to a 32 bits value, used to return the length of the message.
Returns	Status of the read operation (ST25FTM_Msg_Status_t enum): ST25FTM_MSG_OK: the message has been successfully read. ST25FTM_MSG_ERROR: an error occurred. ST25FTM_MSG_BUSY: the message has not been read, must retry.
Description	This function reads the peer device message from the FTM memory.

Table 33. ST25FTM_WriteMsg function

Function	ST25FTM_Msg_Status_t ST25FTM_WriteMsg(uint8_t* msg, uint32_t msg_len)
Parameters	msg: a buffer containing the message to be sent. msg_len: the length of the message.
Returns	Status of the read operation (ST25FTM_Msg_Status_t enum): ST25FTM_MSG_OK: the message has been successfully sent. ST25FTM_MSG_ERROR: an error occurred. ST25FTM_MSG_BUSY: the message has not been written, must retry.
Description	This function writes the message to the FTM memory.

3.3 ST25FTM Internal states

This section describes the different internal states of the ST25FTM states machines. The ST25FTM library has two state machines, one for the transmission and one for reception.

3.3.1 ST25FTM transmission state machine

The ST25FTM library transmission state machine works through the following states.

Table 34. ST25FTM transmission states

ST25FTM_WRITE_IDLE	When the ST25FTM_SendCommand function is called to provide data to be transmitted, the FTM transmission state machine initializes all the required state variables and the CRC IP.
ST25FTM_WRITE_CMD	The ST25FTM transmission state machine prepares the next segment to be sent, according to the acknowledgment scheme selected (doing nothing/computing segment CRC).
ST25FTM_WRITE_SEGMENT	The ST25FTM transmission state machine prepares the next message to be sent, according to its position in the segment.
ST25FTM_WRITE_PKT	The ST25FTM transmission state machine actually writes the message to the FTM memory.
ST25FTM_WRITE_WAIT_READ	The ST25FTM transmission state machine waits until the message has been read by the peer device.
ST25FTM_WRITE_READ_ACK	In case an ACK is expected from peer device, the ST25FTM transmission state machine poll to read the ACK message.
ST25FTM_WRITE_DONE	The transmission has successfully gone to its end.
ST25FTM_WRITE_ERROR	An unrecoverable error occurred.

3.3.2 ST25FTM reception state machine

The ST25FTM library reception state machine works through the following states.

Table 35. ST25FTM reception states

ST25FTM_READ_IDLE	When the ST25FTM_ReceiveCommand function is called to provide data to be transmitted, the FTM reception state machine initializes all the required state variables and the CRC IP.
ST25FTM_READ_CMD	The FTM reception state machine waits for an incoming message from the peer device.
ST25FTM_READ_PKT	The FTM reception state machine reads the available message from the peer device. The message in unpacked and actions are taken according to the protocol metadata: initializing reception state variable, CRC checks, ...
ST25FTM_READ_WRITE_ACK	The FTM reception state machine writes an ACK message to the FTM memory.
ST25FTM_READ_WRITE_NACK	The FTM reception state machine writes a NACK message to the FTM memory.
ST25FTM_READ_WRITE_ERR	The FTM reception state machine writes an ERROR message to the FTM memory.
ST25FTM_READ_WAIT_ACK_READ	The FTM reception state machine waits for the ACK/NACK/ERROR message to be read by a peer device.
ST25FTM_READ_DONE	The reception has successfully completed.
ST25FTM_READ_ERROR	An unrecoverable error occurred.

Revision history

Table 36. **Revision history**

Date	Revision	Changes
18-Jun-2020	1	Initial release
17-Dec-2021	2	Updated: • Section 2 ST25FTM protocol • Section 3.1 ST25FTM API • Section 3.3.1 ST25FTM transmission state machine Added: • Table 20. ST25FTM_CheckError function • Table 21. ST25FTM_IsIdle function • Table 22. ST25FTM_Reset function Deleted: • Table 15. ST25FTM_GetCryptoTime function

Contents

List of tables

List of figures