How to post and receive VDM using STUSB4531?

Introduction

Hybrid mode

The STUSB4531 includes a patented hybrid mode that allows the system to post any message and read any received message by using the I²C read and write operations. The STUSB4531 flags the system by using its ALERT interrupt pin.

1 I²C commands

Two generic functions are defined in this document to indicate I2C read or write actions.

For both functions, the device address is omitted to avoid overloading the document. The device address can be 0x28 or 0x29, as specified in the DS15023: STUSB4531 datasheet.

I2C_Read(*register_address*, *number_of_bytes*):

register_address: indicates the address starting point to be read

number_of_bytes: indicates if a single or multiple consecutive read is performed

I2C_Write(*register_address*, data, *number_of_bytes):*

register_address: indicates the address starting point to be written

data: is the content to be written

number_of_bytes: indicates if a single or multiple consecutive write is performed

Please note that multi-byte registers need to be accessed using the little-endian coding.

2 STUSB4531 prerequisites

As per specified in the USBPD specification [6], most messages must be answered within tReceiverResponse to prevent tSenderReponseTimer expiration. The Sink can start an AMS (Atomic Message Sequence) if the Source authorizes it using collision avoidance mechanism.

2.1 Interrupt setup for sending or receiving a message

Some PD interrupts must be unmasked to notify that:

- A message is allowed to be posted or modified in TX_BUFFER: TX_BUFFER_READY

Unmask PD_STATUS_AL as described in [3].

Some PRL interrupts must be unmasked to notify that:

- Message has been received: PRL_MSG_RECEIVED
- Message has been sent: PRL_MSG_SENT
- Message has been discarded: PRL_TX_DISCARD
- Message has never been acknowledged by counterpart: PRL_TX_ERR

Unmask PRL_STATUS_AL as described in [3].

2.2 Sending a command to STUSB4531 policy engine

To initiate an AMS and send a message to the counterpart, the STUSB4531 must be advised to send the message using the TX_SEND_MSG command.

Upon receipt of certain responses, the STUSB4531 must be informed that no further messages need to be sent using the IGNORE_RX_MSG directive.

As per [2] COMMAND register is used:

Table 1. COMMAND register

7	6	5	4	3	2	1	0
Reserved				CMD[3:0]			
				RW			

Address:	0x0031
Reset:	0x00
[3:0]	**CMD:** Software command: – 0x1: (TX_SEND_MSG) request the send of a message already present in the Tx buffer registers. – 0x2: (IGNORE_RX_MSG) ignore received message (Tx buffer discarted). – 0x9: (BYPASS_AMS) force Send message without checking TxOK.

3 Structured VDM

As per [6], the Discovery_Identity command request must be sent to identify the port partner.

Both source and sink can initiate a structured VDM AMS. However, the UFP shall not initiate Enter Mode or Exit Mode commands.

3.1 Swapping data role

3.1.1 Tx Byte count

Tx Byte count is defined as the number of bytes, including the Header to be transmitted.

The DR_SWAP message is a control message so there is no data object.

Tx Byte count is therefore 2 bytes for the Header and 0 bytes for the data object:

@TX_BYTE_CNT = 0x02

3.1.2 Specification Revision

Specification revision is established during the first explicit contract and remains static until disconnection occurs.

- Rx_buffer[15:0] = I2C_Read(RX_HEADER,2)
- Spec_rev = Rx_buffer[4:6]

3.1.3 Port Data Role

As per [6], the Sink is UFP by default. However, the data role might have changed before the MCU is up and running, depending on the STUSB4531 NVM settings.

Port data role can be determined by checking the @PD_STATUS.DATA_ROLE register.

3.1.4 DR_SWAP header

As per [6], message header construction for DR_SWAP is:

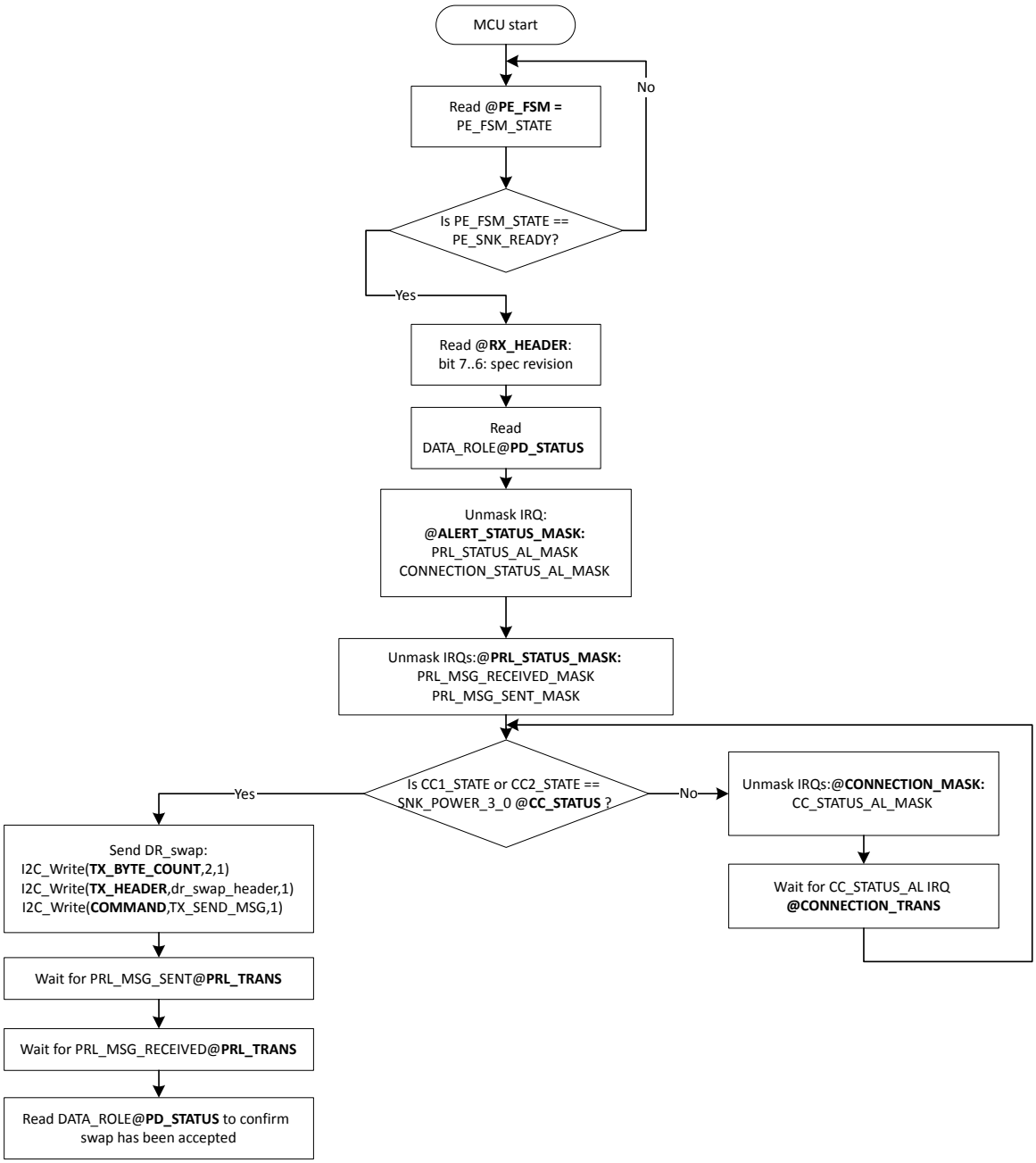
Table 2. DR_swap message header

Bit(s)	Field name	Value	Comment
15	Extended	0	Control or data message
14...12	Number of Data Objects	0	Control message
11...9	MessageID	Handled by hardware	
8	Port Power Role	0	Sink
7..6	Specification Revision	Section 3.1.2: Specification Revision	
5	Port Data Role	Section 3.1.3: Port Data Role	
4..0	Message Type	01001	DR_SWAP message code

TX_HEADER register needs to be filled with above DR_Swap header.

3.1.5 Sending DR_Swap flow chart

Figure 1. **DR_Swap flow chart**



3.2 Discover Identity Request

3.2.1 Tx Byte count

The Tx Byte count is defined as the number of bytes, including the Header to be transmitted.
The Discovery Identity Request command message is a data message with single 32-bit data object (4 bytes).
Tx Byte count is then 2 bytes for the Header and 4 bytes for the data object:
@TX_BYTE_CNT = 0x06
I2C_Write(TX_BYTE_CNT, 0x06,1)

3.2.2 TX_Header

Table 3. Discovery Identity Request message Header

Bit(s)	Field name	Value	Comment
15	Extended	0	Control or data message
14...12	Number of Data Objects	1	Single 32-bits data object
11...9	MessageID	Handled by hardware	
8	Port Power Role	0	Sink
7..6	Specification Revision	Section 3.1.2: Specification Revision	
5	Port Data Role	Section 3.1.3: Port Data Role	
4..0	Message Type	01111b	Vendor Defined Message code

3.2.3 TX Data object

The Discovery Identity Request message has a single data object as for the VDM Header.

Table 4. VDM Header for Discovery Identity

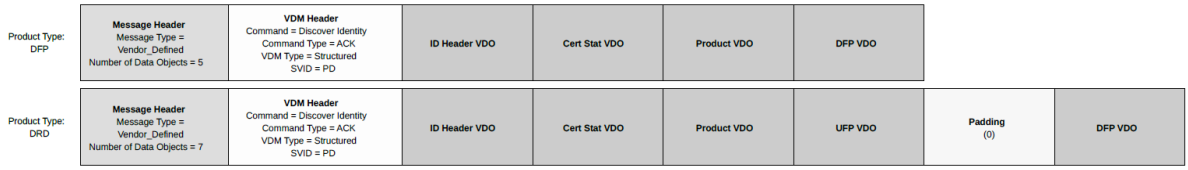
Bit(s)	Field name	Value	Comment
31..16	Standard or Vendor ID (SVID)	0xFF00	PD SID allocated for USBPD spec
15	VDM Type	1b	Structured VDM
14..13	SVDM Version (Major)	01b	As per [6]
12..11	SVDM Version (Minor)	01b	As per [6]
10..8	Object Position	0b	As per [6]
7..6	Command Type	00b	REQ: Request from Initiator Port
5	Reserved	0b	
4..0	Command	00001b	Discovery Identity

3.3 Discovery Identity Acknowledge message

As per [6], the Source might acknowledge the Discovery Identity Request by sending its ID Header VDO along with Cert Stat VDO, product VDO and DFP VDO. UFP VDP could also be listed if Source is a DRD device.

Depending on the product type, the message received is as follows:

Figure 2. Discovery Identity response format from USBPD specification



3.3.1 Message reception length

Nb_of_bytes = I2C_Read(@RX_BYTE_COUNT, 1)

- If the product type is DFP: There is one Header and five data objects: Nb_of_bytes is 22.
- If the product type is DRD: There is one Header and seven data objects: Nb_of_bytes is 30.
- If Nb_of_bytes is 2: most likely, a Not_Supported message has been received.

3.3.2 Discovery Identity response Header (RX_BUFFER)

Table 5. Discovery Identity Request message Header

Bit(s)	Field name	Value	Comment
15	Extended	0	Control or data message
14...12	Number of Data Objects	5 or 7	Depending if Product Type is DFP or DRD
11...9	MessageID	Handled by hardware	
8	Port Power Role	1	Source
7..6	Specification Revision	Section 3.1.2: Specification Revision	
5	Port Data Role	Section 3.1.3: Port Data Role	
4..0	Message Type	01111b	Vendor Defined Message code

3.3.3 Discovery Identity data objects response (RX_DATA_OBJ_224BITS)

Register needs to be accessed through a multi-read from the base address.

Disco_resp = I2C_Read(RX_DATA_OBJ_224BITS, Nb_of_bytes-2)

Message interpretation must follow [6]

3.4 Discover SVID command request

Like the Discovery Identity request command, the Discover SVID request is a total of 6 bytes. The Section 3.1.1: Tx Byte count section must be followed.

3.4.1 TX_header

Table 6. Discover SVIDs Request message Header

Bit(s)	Field name	Value	Comment
15	Extended	0	Control or data message
14...12	Number of Data Objects	1	Single 32-bits data object
11...9	MessageID	Handled by hardware	
8	Port Power Role	0	Sink
7..6	Specification Revision	Section 3.1.2: Specification Revision	
5	Port Data Role	Section 3.1.3: Port Data Role	
4..0	Message Type	01111b	Vendor Defined Message code

3.4.2 TX Data object

The Discovery SVIDs Request message has a single data object as for the VDM Header.

Table 7. VDM Header for Discovery SVID

Bit(s)	Field name	Value	Comment
31..16	Standard or Vendor ID (SVID)	0xFF00	PD SID allocated for USBPD spec
15	VDM Type	1b	Structured VDM
14..13	SVDM Version (Major)	01b	As per [6]
12..11	SVDM Version (Minor)	01b	As per [6]
10..8	Object Position	0b	As per [6]
7..6	Command Type	00b	REQ: Request from Initiator Port
5	Reserved	0b	
4..0	Command	00010b	Discovery SVIDs

3.4.3 Discovery SVIDs response Header (RX_BUFFER)

Table 8. Discovery Identity Request message Header

Bit(s)	Field name	Value	Comment
15	Extended	0	Control or data message
14...12	Number of Data Objects	5 or 7	Depending if Product Type is DFP or DRD
11...9	MessageID	Handled by hardware	
8	Port Power Role	1	Source
7..6	Specification Revision	Section 3.1.2: Specification Revision	
5	Port Data Role	Section 3.1.3: Port Data Role	
4..0	Message Type	01111b	Vendor Defined Message code

3.4.4 Discover SVIDs response length

Nb_of_bytes = I2C_Read(@RX_BYTE_COUNT, 1)

The message header is 2 bytes.

The VDM header is one 32-bits data object (4 bytes).

According to [6], SVIDs are returned two per VDO (one 32-bit data object for two VDOs or one VDO and null padding).

The maximum number of supported VDOs is calculated as: (Nb_of_bytes – 6) / 2

3.4.5 Discover SVIDs data objects response (Rx_DATA_OBJ_224BITS)

The register needs to be accessed through a multi-read from the base address.

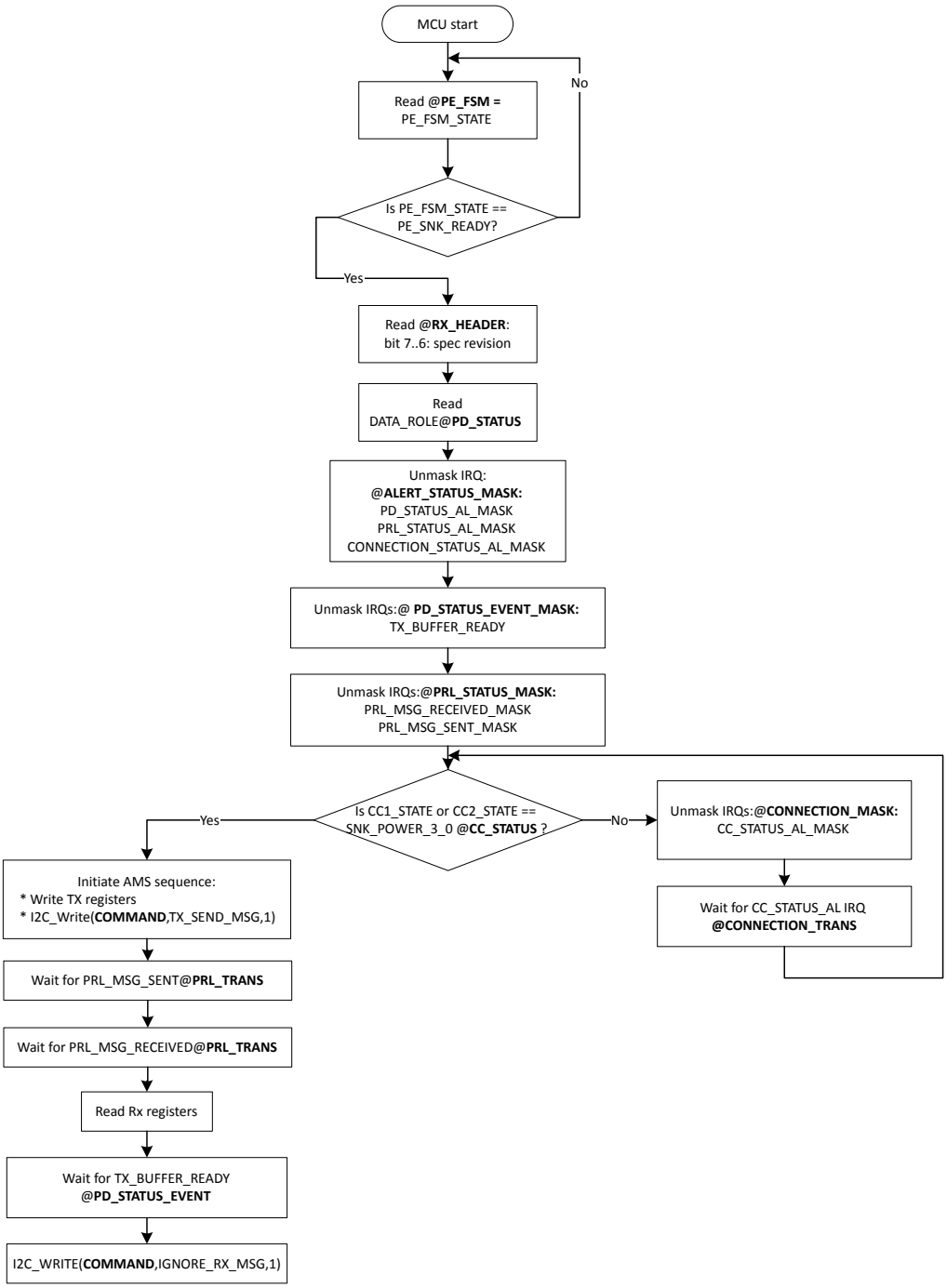
Disco_response = I2C_Read(RX_DATA_OBJ_224BITS, Nb_of_bytes - 2)

Message interpretation needs to follow [6]

4 Handling VDM AMS initiated by STUSB4531

Both the Discover Identity Request and the Discover SVID command request implementation must follow the flow in Figure 3.

Figure 3. Initiate VDM AMS flow



5 Handling VDM AMS sequence initiated by Source

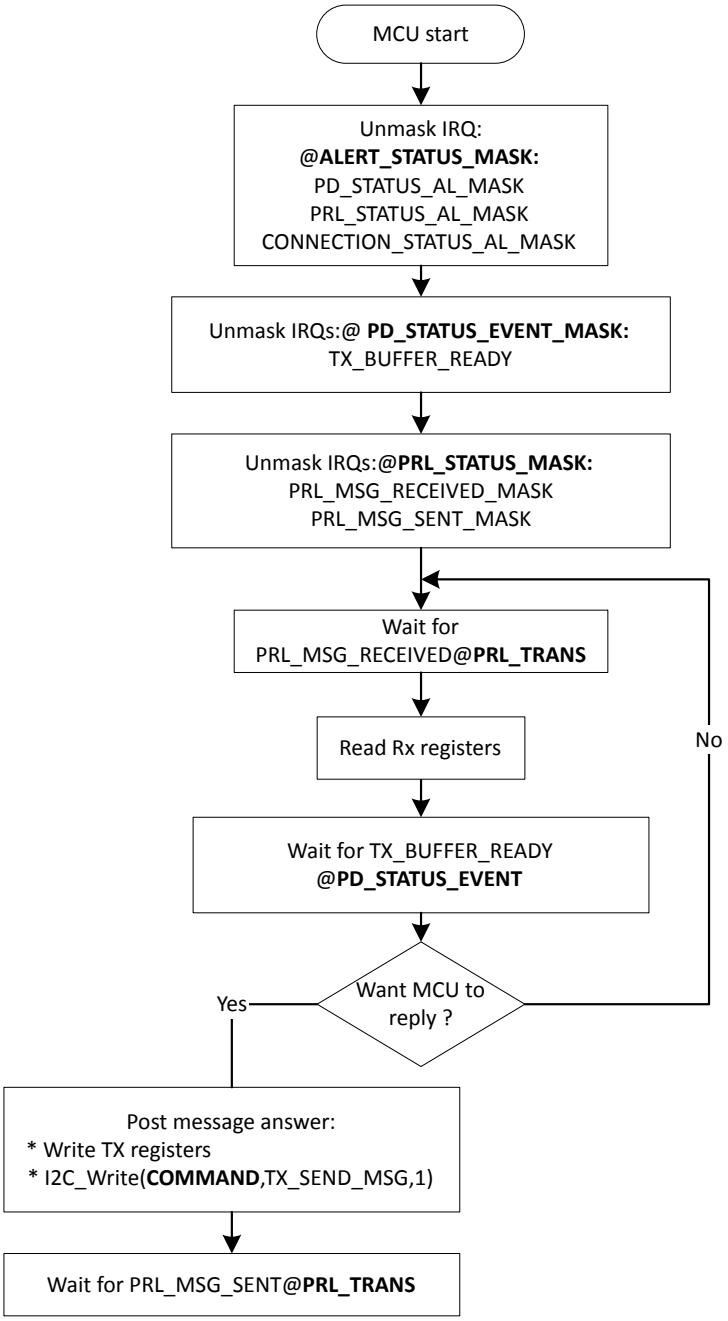
When receiving a message, the following information can be extracted through the RX_HEADER register to prepare the response.

Table 9. RX_HEADER

Bit(s)	Field name	Description
15	Extended	0: Control or data message 1: Extended message
14...12	Number of Data Objects	0: control message
7..6	Specification Revision	
5	Source Port Data Role	Not (STUSB4531 data role)
4..0	Message Type	

When receiving an AMS initiated by the Source, the Message Type must be decoded using [6].

Figure 4. VDM AMS initiated by Source



6 References

[1] DS15023: STUSB4531 datasheet

[2] RM0562: STUSB4531 Register Map

[3] AN6406: How to handle STUSB4531 interrupts

[4] AN6466: How to identify the current explicit contract using STUSB4531?

[5] STSW-STUSB020: Graphical User Interface for STUSB4531

[6] USB Power Delivery specification: https://www.usb.org/document-library/usb-power-delivery

[7] USB type-C Cable and Connector specification: https://www.usb.org/document-library/usb-type-cr-cable-and-connector-specification-release-25

Revision history

Table 10. **Document revision history**

Date	Revision	Changes
10-Jun-2026	1	Initial release.

Contents

List of tables

List of figures