



**CDI2**

# **Simplified CDI2 Device Implementation**

**Design Study & Reference API**

**Author: Peter.Schurek@ait.ac.at**

**Version 1.5 / 29.09.2023**

Version	Date	Change History/Reason	Name
1.0	24.01.2023	1 <sup>st</sup> version for internal review	Schurek Peter
1.1	25.01.2023	Missing shutdown chapter added, test concept added	Schurek Peter
1.2	26.01.2023	Results of internal review	Schurek Peter
1.3	27.01.2023	Update after customer presentation	Schurek Peter
1.4	09.08.2023	Update with learnings from implementation	Schurek Peter
1.5	29.09.2023	Updated document name and references	Venditti Benjamin

## Table of Contents

1. Introduction .....	5
2. References.....	6
2.1. Terms and abbreviations .....	7
3. Logic and Electrical Interfaces.....	10
3.1. TTS.....	10
3.2. RMII .....	10
3.3. PPP:UART .....	11
3.4. CDI2:UART .....	11
4. Software Interface.....	12
4.1. CDI2:UART .....	12
4.1.1. PDU Transmission and Reception.....	12
4.1.2. Startup Phase.....	15
Software Detector Reset 0x10.....	16
Protocol Version 0x11.....	16
Protocol Version Answer 0x21 .....	17
TTS Reset 0x23 .....	18
NMT Reset 0x24.....	18
4.1.3. Initialisation Phase.....	19
Start CDI2 0x15 .....	20
Start CDI2 Answer 0x25 .....	21
DET Status 0x16 .....	21
BSM Status 0x26 .....	21
4.1.4. Feed-Off Phase .....	22
Set Maintenance State 0x17.....	22
Set Error State 0x18.....	23
4.1.5. Sorting Phase .....	23
Banknote ID 0x81.....	25
Banknote Trigger 0x80.....	26
Banknote Info 0x82.....	26
Banknote Recognition 0x42.....	26
Banknote Result 0x41 .....	27
4.1.6. Shutdown Phase .....	27
4.1.7. Error Handling and Logging .....	28
FATAL 0x29 .....	30
ERROR 0x2A .....	30
WARNING 0x2B.....	30
INFO 0x2C, DEBUG 0x2D, TRACE 0x2E.....	31
4.1.8. SW Update .....	31
Prepare Update 0x87.....	32
Accept Update 0x47, Reject Update 0x48.....	33
Perform Update 0x88 .....	33
4.2. PPP:UART .....	33
4.3. Test Concept .....	34

5. License Note .....	35
-----------------------	----

## 1. Introduction

The CDI2 Device Interface Core implements an embedded middleware system on an FPGA that connects a CDI2 detector application logic to a standard CDI2 interface provided by a BSM. It hides the intricacies of the CDI2 Powerlink layer configuration, implementation, and interaction with the CDI2 device state machine [Ref 1.] behind a much simpler serial interface that is a proprietary extension of the earlier CDI standard [Ref 2.].

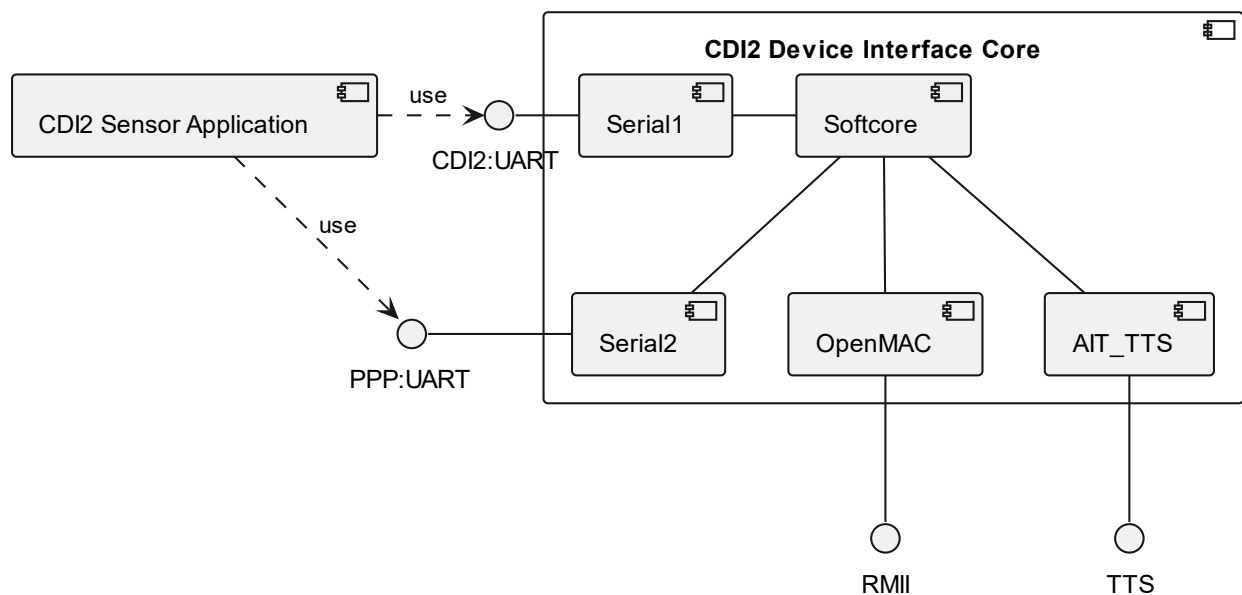


Figure 1: CDI2 IP Core Block Diagram

The CDI2 IP Core comprises a processor subsystem that executes the PowerLink and CDI2 layer services, 2 serial interfaces (standard 16550 register compatible IP cores [Ref 10.]), the OpenMAC IP core supporting the real-time response requirements of the DMB, and the device subset of the TTS interface logic developed by AIT for the CDI2 simulators.

The intended baud rate of the CDI2:UART (1.25 Mbaud/s with 8N1) adds a transmission latency of 8µs/Byte. This transmission overhead must be considered for the timing budget the CDI2 standard [Ref 1.] allocates for BNRESULT messages (45ms). The design assumption is that a detector application transmits no more than one BNRESULT segment per banknote and supplies about 100 bytes (unescaped) of supplemental data.

### RMII

The Media Independent Interface connects the OpenMAC IP core to a suitable PHY (e.g., the DP83822IRHBR from Texas Instruments that is used by the CDI2 simulator [Ref 4.]). Alternative PHY chips supported by the OpenPowerLink stack are the Micrel KSZ8051RNL and the Marvell 88E1111.

### TTS

The (optional) TTS interface provides three input signals, RESET request, Banknote Present (BP), and Transport Clock (TC). A READY output is used to indicate the device status to the BSM.

**CDI2:UART**

The Application controls (start, stop and reset/restart) and monitors the CDI2 Device Interface Core via this serial link by exchanging CDI2 protocol data units (PDU) defined later in this document. PDU exchanges at certain points in the CDI2 state machine allow the application to perform the CDI2 device initialization and configuration information exchanges with the BSM. In the sorting phase, banknote information and results are received and sent over this link.

**PPP:UART**

CDI2 detectors must implement FTP (used during DMB initialization and SW update) and HTTP/S servers [Ref 1.] that are accessed by the BSM over the DMB. The packets of the underlying IP protocol are wrapped into Async frames of the Powerlink and transferred in the asynchronous phase of the PowerLink cycle. The CDI2 Device Interface Core implements a PPP/DMB bridge interface that allows the detector application to connect its servers to a standard PPP network interface [Ref 5.][Ref 6.] on its side.

## 2. References

[Ref 1.]

Common Detector Interface 2 (CDI2) Specifications v2.7C, 21 March 2022

[https://www.ecb.europa.eu/euro/pdf/CDI2\\_version\\_2.7C\\_Final.pdf](https://www.ecb.europa.eu/euro/pdf/CDI2_version_2.7C_Final.pdf)

[Ref 2.]

Common Detector Interface (CDI) Specifications v1.0, Revision B, 27 May 2013

[https://www.ecb.europa.eu/euro/pdf/Common\\_Detector\\_Interface\\_CDI\\_Spec\\_1\\_0\\_rev\\_B.pdf](https://www.ecb.europa.eu/euro/pdf/Common_Detector_Interface_CDI_Spec_1_0_rev_B.pdf)

[Ref 3.]

OpenMAC Toplevel Documentation

[https://github.com/OpenAutomationTechnologies/openPOWERLINK\\_V2/blob/master/hardware/ipcore/doc/openmac/md/openmac.md](https://github.com/OpenAutomationTechnologies/openPOWERLINK_V2/blob/master/hardware/ipcore/doc/openmac/md/openmac.md)

[Ref 4.]

DP83822 Robust, Low Power 10/100 Mbps Ethernet Physical Layer Transceiver

<https://www.ti.com/lit/gpn/dp83822i>

[Ref 5.]

The Point-to-Point Protocol (PPP)

<https://www.rfc-editor.org/rfc/rfc1661>

[Ref 6.]

The PPP Internet Protocol Control Protocol

<https://www.rfc-editor.org/rfc/rfc1332.html>

[Ref 7.]

FreeRTOS™ Real-Time Operating System for Microcontrollers

<https://www.freertos.org/index.html>

[Ref 8.]

Quantum Leaps State Machine & Tools for Embedded Systems

<https://www.state-machine.com>

[Ref 9.]

LwIP – A Lightweight TCP/IP Stack

<https://savannah.nongnu.org/projects/lwip/>

[Ref 10.]

PC16550D Universal Asynchronous Receiver/Transmitter With FIFOs

<https://web.archive.org/web/20180826215135/http://www.ti.com/lit/ds/symlink/pc16550d.pdf>

[Ref 11.]

fwuart-16550

<https://github.com/Featherweight-IP/fwuart-16550>

[Ref 12.]

Mars ST3 Base Board for FPGA Modules

<https://www.enclustra.com/en/products/base-boards/mars-st3>

[Ref 13.]

Mars ZX2 Xilinx Zynq 7010/7020 All Programmable SoC Module

<https://www.enclustra.com/en/products/system-on-chip-modules/mars-zx2/>

[Ref 14.]

PPP in HDLC-Like Framing

<https://www.rfc-editor.org/rfc/rfc1662.html>

[Ref 15.]

An Ethernet Address Resolution Protocol

<https://www.rfc-editor.org/rfc/rfc826.html>

## 2.1. Terms and abbreviations

<i>ACFC</i>	Address and Control Field Compression
<i>AIT</i>	Austrian Institute for Technology
<i>ARP</i>	Address Resolution Protocol
<i>ASCII</i>	American Standard Code for Information Interchange
<i>ASCIZ</i>	ASCII Zero
<i>BFA</i>	Belt-Free Area
<i>BN</i>	Banknote
<i>BP</i>	Banknote Present
<i>BSM</i>	Banknote Sorting Machine
<i>CDI</i>	Common Detector Interface

<i>CDI2</i>	Common Detector Interface 2
<i>CN</i>	Controlled Node
<i>CRC</i>	Cyclic Redundancy Check
<i>DMB</i>	Detector Machine Bus
<i>FPGA</i>	Field Programmable Gate Device
<i>FTP</i>	File Transfer Protocol
<i>HDLC</i>	High-level Data Link Control
<i>HPS</i>	Hard Processor System
<i>HTTP/S</i>	Hyper Text Transfer Protocol
<i>HW</i>	Hardware
<i>IP</i>	Internet Protocol
<i>IP core</i>	Intellectual Property Core
<i>LCP</i>	Link Control Protocol
<i>LSB</i>	Least Significant Byte
<i>LwIP</i>	Lightweight IP
<i>MAC</i>	Medium Access Layer
<i>MII</i>	Media Independent Interface
<i>MN</i>	Managing Node
<i>NMT</i>	Node Management
<i>MRU</i>	Maximum Receive Unit
<i>MSB</i>	Most Significant Byte
<i>NCP</i>	Network Control Protocol
<i>PDU</i>	Protocol Data Unit
<i>PFC</i>	Protocol Field Compression
<i>PHY</i>	Physical interface
<i>PPP</i>	Point-to-Point Protocol
<i>RMII</i>	Reduced Media Independent Interface



<i>RX</i>	Receive, Receiver
<i>SDO</i>	Synchronous Data Object
<i>SoC</i>	System on a Chip
<i>SW</i>	Software
<i>TBD</i>	to be defined
<i>TC</i>	Transport Clock
<i>TCP</i>	Transmission Control Protocol
<i>TTS</i>	Transport Timing Signals
<i>TX</i>	Transmit, Transmitter
<i>UART</i>	Universal Asynchronous Receiver-Transmitter

### 3. Logic and Electrical Interfaces

These are FPGA level logic interfaces provided by the IP core. TTS and MII interface signals are expected to be assigned to FPGA pins connected to electrical interfaces while the UART signals might as well be connected to detector logic implemented on the same FPGA.

#### 3.1. TTS

The electrical specification (RS-422 and current loop), connector type (DE-15S), and pinout of the TTS is detailed in the CDI2 standard [Ref 1.], on FPGA side the signals are expected to be connected to single-ended GPIO pins:

Signal Name	Direction	Implemented as
TTS_BP (Banknote Present)	In	RS-422
TTS_TC (Transport Clock)	In	RS-422
TTS_RESET	In	Current loop
TTS_READY	Out	Current loop

Table 1: TTS Signals

The signal lines must be galvanically isolated to prevent electrical damage to BSM or detector. The BSM supplies +5V on pin 11 to power opto-couplers on the detector board.

#### 3.2. RMII

The FPGA implements the MII interface with single-ended GPIO pins, the PHY chip, bootstrap pull-up/downs, magnetics and RJ45 registered jack must be provided by the board implementation.

Signal Name	Direction
smi_nPhyRst (PHY reset#)	Out
smi_clk (management clock)	Out
smi_dio (management data)	InOut
rmii_rxCrsDataValid (Carrier Sense and RX data valid)	In
rmii_rxData[2] (RX data)	In
rmii_rxError (receive error)	In
rmii_txEnable (TX enable)	Out
rmii_txData[2] (TX data)	Out

Table 2: RMII Signals

### 3.3. PPP:UART

The PPP UART should be a 16550 register compatible IP core provided either as stock IP core by the FPGA vendor or, if the limited FIFO depth or the licensing model proves problematic, a free implementation from e.g., OpenCore will be evaluated [Ref 11.]. The UART must provide an internal FIFO buffer for performance reasons. The frame format uses 1 start, 8 data, and 1 stop bit, no parity (8N1). Thus, to achieve the necessary 1Mbps transmission rate, the supported baud rate of the serial line code must be 1.25 Mbaud/s.

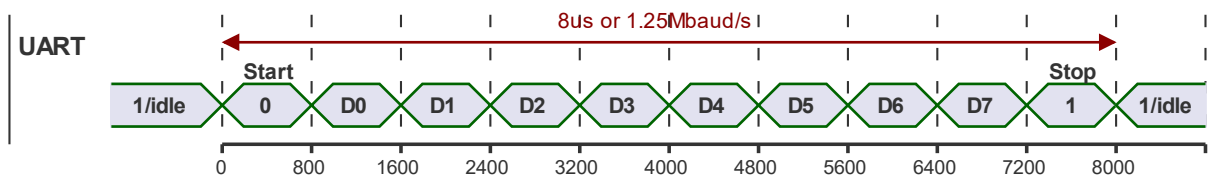


Figure 2: UART Waveform

Signal Name	Direction
ppp_rxd (receive data)	In
ppp_txd (transmit data)	Out

Table 3: Minimal PPP:UART Signals

Xilinx offers a 16550 compatible AXI UART core as part of the Vivado license. Intel/Altera offers a lightweight Avalon UART core with the Quartus license, a full implementation with an internal FIFO needs a separate license. The Featherweight UART sources come with an Apache license [Ref 11.].

### 3.4. CDI2:UART

The CDI2 UART is configured the same as the PPP UART (1.25 Mbaud/s, 8N1).

Signal Name	Direction
cdi2_rxd (receive data)	In
cdi2_txd (transmit data)	Out

Table 4: Minimal CDI2:UART Signals

## 4. Software Interface

### 4.1. CDI2:UART

#### 4.1.1. PDU Transmission and Reception

To detect and recover safely and robustly from transmission errors all CDI2 PDUs are framed by unique start and stop characters which must be escaped in the PDU. This link layer is based on the CDI standard, thus paraphrasing [Ref 2.]:

There is a set of special (reserved) characters, namely 0xF8 to 0xFE. The following special characters are functionally defined:

0xF8	reserved
0xF9	reserved
0xFA	reserved
0xFB	reserved
0xFC	CDI_STX (start of frame)
0xFD	CDI_ETX (end of frame)
0xFE	CDI_ESC (escape)

**Table 5: Special Characters**

Special characters appear unescaped in the data stream only if their special function is intended. All data characters xx that equal a special character are translated/escaped into a two-byte sequence CDI\_ESC yy, where yy is the one's complement of xx. All other characters remain as they are. An N+1 byte long CDI2 PDU:

ID	Byte0	Byte1	...	ByteN
----	-------	-------	-----	-------

is escaped, then framed as follows:

CDI_STX	ID	TByte0	TByte1	...	TByteN	CRC-16	CDI_ETX
---------	----	--------	--------	-----	--------	--------	---------

where TByte0 ... TByteN are the escaped versions of Byte0 ... ByteN versions as described above. The CRC-16 field is the standard CRC-CCITT (CRC-16) defined by generator polynomial  $0x1021 = x^{16} + x^{12} + x^5 + 1$  with initial value 0xFFFF. It is calculated over the unescaped PDU (the ID, Byte0...ByteN sequence) and appended in LSB first order before escaping:

CDI2 PDU		0x84	0xFA				
CRC-16 appended (0x8406)		0x84	0xFA	0x06	0x84		
Escaped		0x84	0xFE	0x05	0x06	0x84	
Framed	0xFC	0x84	0xFE	0x05	0x06	0x84	0xFD

**Figure 3: CDI2 PDU CRC Calculation, Escaping, and Framing**

Byte order in the CDI2 PDU is little endian (LSB transmitted first). All PDUs start with a command or response ID byte.

The CDI\_STX/CDI\_ETX framing together with the CRC-16 allow both sides to detect transmission errors. Regarding dealing with detected errors, [Ref 2.] states: “In case of any interface error the BSM cannot start the operation and shall inform the operator about the interface problems with an understandable error message on the operator screen.” To support this strategy, the CDI2 IP Core will, when recognizing any link layer errors on its RX side, attempt to resynchronize the link by clearing the receive FIFO until it becomes empty or a CDI\_STX frame start byte is received, then send an ERROR message to the application. If the application detects a link error, it should resynchronize the link in a similar way (drop characters until the link becomes idle or a proper CDI\_STX byte is received). After learning the current CDI2 states with a Protocol Version/Protocol Version Answer handshake, it can then decide to request a transition to DS\_ERROR, just reset the CDI2 IP Core or just report the link layer error as warning.

Command/Response ID	Full Name	Direction	Length <sup>1</sup>	Operation Phase
0x10	Software Detector Reset	App→Core	6	Startup
0x11	Protocol Version	App→Core	7	
0x21	Protocol Version Answer	Core→App	13	
0x12	Machine Info (reserved)			
0x22	Detector Info (reserved)			
0x23	TTS Reset	Core→App	5	
0x24	NMT Reset	Core→App	6	
0x15	Start CDI2	App→Core	8	
0x25	Start CDI2 Answer	Core→App	5	
0x16	DET Status	App→Core	6	
0x26	BSM Status	Core→App	6	
0x17	Set Maintenance State	App→Core	6	
0x18	Set Error State	App→Core	6	
0x29	FATAL	Core→App	6+N	
0x2A	ERROR	Core→App	6+N	
0x2B	WARNING	Core→App	6+N	
0x2C	INFO	Core→App	6+N	
0x2D	DEBUG	Core→App	6+N	
0x2E	TRACE	Core→App	6+N	
0x80	Banknote Trigger	Core→App	10	Sorting
0x81	Banknote ID	Core→App	21	
0x82	Banknote Info	Core→App	12	
0x83	Banknote ID and Info (reserved)			
0x41	Banknote Result	App→Core	13+N	
0x42	Banknote Recognition	App→Core	12	
0x84	Feed Off (reserved)			
0x44	Feed Off Answer (reserved)			
0x85	Feed On (reserved)			Feed Off
0x45	Feed On Answer (reserved)			
0x86	Raw Data Request (reserved)			
0x46	Raw Data Answer (reserved)			
0x47	Accept Update	App→Core	5	
0x48	Reject Update	App→Core	5	
0x87	Prepare Update	Core→App	5	
0x88	Perform Update	Core→App	5	

Table 6: CDI2 PDU Overview

<sup>1</sup> Unescaped, but includes STX, CRC, and ETX

### 4.1.2. Startup Phase

After a power-up reset or a reset requested by the application via the Software Detector Reset command, the CDI2 IP core indicates its readiness to start communication by sending an unsolicited Protocol Version Answer message with status byte set to 0x03 (Detector Startup).

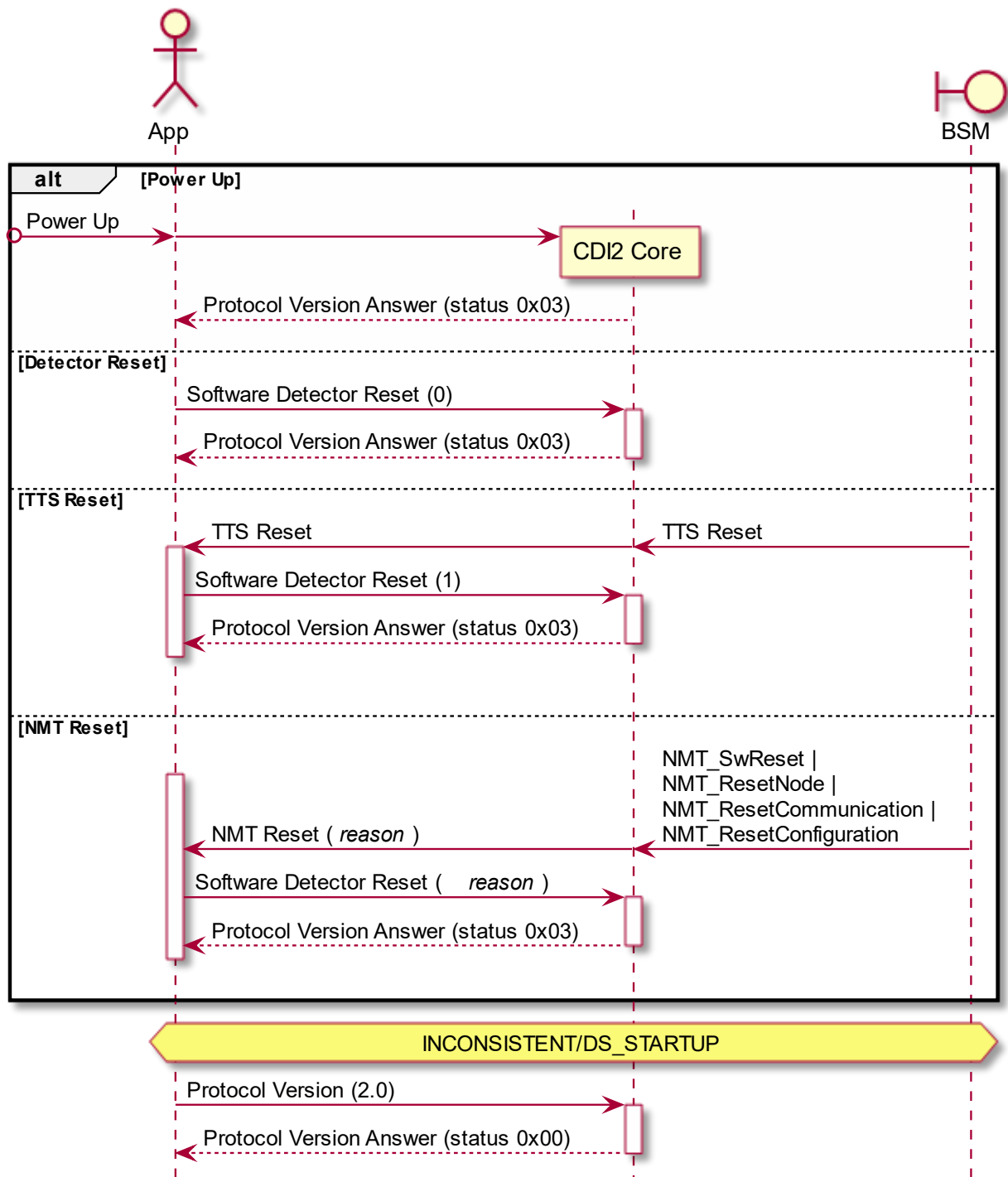


Figure 4: CDI2 IP Core Startup

The protocol version field indicates the highest and lowest serial protocol version supported over the serial interface with MSB and LSB interpreted as major and minor version numbers following a simple semantic versioning scheme. Thus, a version number 0x0203 indicates a semantic version “2.3” promising compatibility with all “2.x” interface protocols up to and including minor version 3 (i.e., 2.0, 2.1, 2.2, and 2.3).

### Software Detector Reset 0x10

The application requests a software restart of the CDI2 IP core. The core restarts and resets all internal states to initial values: CDI2 state is DS\_STARTUP, maintenance and error state is cleared (0), TTS Ready is low, and Powerlink is stopped. When ready to communicate, the core sends a Protocol Version Answer message with status byte 0x3 (Detector Start-Up) and indicating its highest supported serial protocol version.

Offset	Length	Code/Data	Description
0	1	0x10	ID byte
1	1		Reset Reason Code (Table 7)

Reset Reason Codes	Description
0x00	Application Reset
0x01	TTS Reset
0x10	NMT_SwReset
0x11	NMT_ResetNode
0x12	NMT_ResetCommunication
0x13	NMT_ResetConfiguration

**Table 7: Reset Reason Codes**

### Protocol Version 0x11

The Application initiates the protocol version negotiation handshake with this command indicating the highest supported serial protocol version described above. If the core answers with a compatible version the application can start CDI2 operation, otherwise the application should stop and inform the operator about the communication problem.

Offset	Length	Code/Data	Description
0	1	0x11	ID byte
1	1		Protocol Version LSB (minor version)
2	1		Protocol Version MSB (major version)



## Protocol Version Answer 0x21

The Application initiates the protocol version negotiation handshake with this command indicating the highest supported serial protocol version described above. If the core answers with a compatible version the application can start CDI2 operation, otherwise the application should stop and inform the operator about the communication problem.

Offset	Length	Code/Data	Description
0	1	0x21	ID byte
1	1		CDI2 IP Core Status 0 ... OK 1 ... Not Ready (temporarily) 2 ... Wrong Protocol 3 ... IP Core Start Up 4 ... reserved 5 ... reserved 6 ... HW Error 7 ... Self-Test failed >=8 ... reserved
2	1	0x00	Protocol Version LSB (minor version)
3	1	0x20	Protocol Version MSB (major version)
4	1		NMT Status Code (Table 8)
5	1		Last known BSM State (Table 9)
6	1		Current CDI2 Device State (Table 9)
7	1		CDI2 Maintenance State [Ref 1.] Bit [0] ... air pulse required Bit [1] ... manual cleaning required Bit [2] ... self-test required Bit [3] ... calibration required Bit [7:4] ... reserved
8	1		CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved

NMT Status Code	Description
0x00	NMT_GS_OFF
0x19	NMT_GS_INITIALIZING
0x29	NMT_GS_RESET_APPLICATION
0x39	NMT_GS_RESET_COMMUNICATION
0x79	NMT_GS_RESET_CONFIGURATION
0x1C	NMT_CS_NOT_ACTIVE
0x1D	NMT_CS_PRE_OPERATIONAL_1
0x4D	NMT_CS_STOPPED
0x5D	NMT_CS_PRE_OPERATIONAL_2
0x6D	NMT_CS_READY_TO_OPERATE
0xFD	NMT_CS_OPERATIONAL
0x1E	NMT_CS_BASIC_ETHERNET
0xFF	NMT_STATE_INVALID

Table 8: NMT Status Codes

BSM State	ID	CDI2 Device State	ID
BS_START_UP	1	DS_START_UP	1
BS_INITIALISATION	2	DS_INITIALISATION	2
BS_FEED_OFF	3	DS_INITIALISED	3
BS_REQUEST_TO_SORT	4	DS_FEED_OFF	4
BS_SORTING	5	DS_READY_TO_SORT	5
BS_REQUEST_TO_SHUT_DOWN	6	DS_SORTING	6
BS_SHUTDOWN	7	DS_READY_TO_SHUT_DOWN	7
BS_ERROR	10	DS_SHUTDOWN	8
		DS_ERROR	10

Table 9: CDI2 Device and BSM Status Codes [Ref 1.]

### TTS Reset 0x23

The core sends this unsolicited message when it detects that the BSM pulled the TTS RESET line high for longer than 100µs [Ref 1.]. It is up to the application to decide whether to perform a full power-up reset or request a Software Detector Reset (0x01).

Offset	Length	Code/Data	Description
0	1	0x23	ID byte

### NMT Reset 0x24

The core sends this unsolicited message when the BSM issues one of the NMT reset commands to the CN. The specific command is passed to the application via the reset reason code (Table 7).

Offset	Length	Code/Data	Description
0	1	0x24	ID byte
1	1		Reset Reason Code (Table 7)

#### 4.1.3. Initialisation Phase

After the application has successfully established and synchronized the control link with the CDI2 IP Core, the CDI2 layer processing must be started with the Start CDI2 command. This also activates the PPP link. At some point, the BSM will enter the synchronous DMB phase and broadcast BSMINFOs carrying BSM status information. The CDI2 IP Core informs the Application about BSM status changes with unsolicited BSM Status messages. The application in turn responds by updating its CDI2 Device State with DET Status commands, thus driving the system state machine.

BSM and application exchange system and device configuration files via FTP [Ref 1.], this communication is performed over the PPP link.

At any point the application can query the current state of the CDI2 IP Core, CDI2 Device layer and NMT state machine with the Protocol Version/Protocol Version Answer handshake.

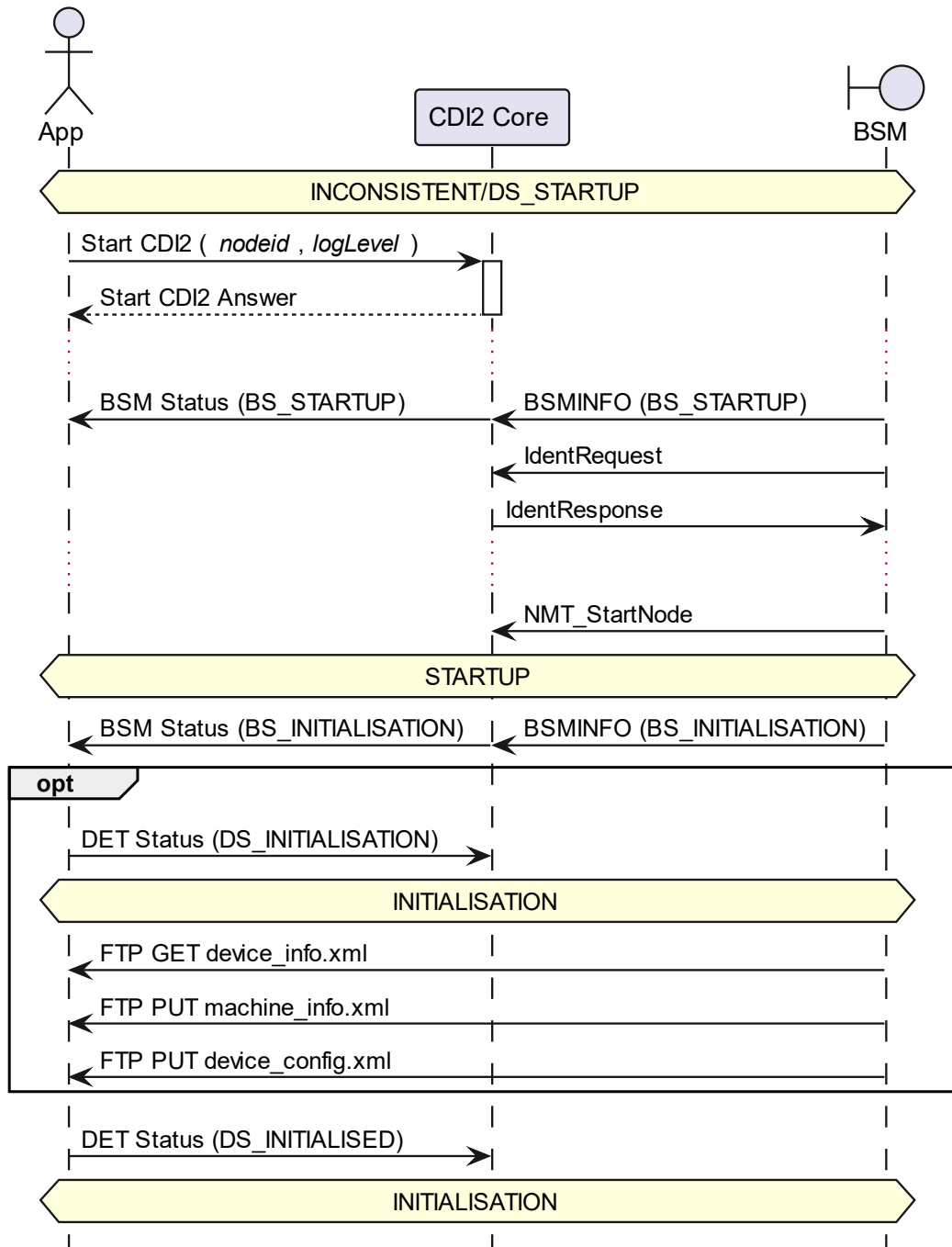


Figure 5: CDI2 IP Core Initialization

## Start CDI2 0x15

This command starts the PPP and Powerlink interfaces of the CDI2 IP core. After joining the synchronous DMB phase as CN with the provided node ID, the application will drive the CDI2 device state machine with DET Status commands in response to unsolicited BSM status messages.

Offset	Length	Code/Data	Description
0	1	0x15	ID byte
1	1		Node ID [Ref 1.]
2	1	0x00	Use 2 <sup>nd</sup> BFA
3	1	0x00	Use TTS interface signals
4	1	0x14	Log/Verbosity Level (Table 10)
5	6	0:0:0:0:0:0	Override default MAC address

Log Level	Description
10	FATAL
20	ERROR
30	WARN
40	INFO
50	DEBUG
60	TRACE

Table 10: CDI2 IP Core Log Levels

If no override MAC address is specified in the Start CDI2 command, a default MAC address is generated for PowerLink interface: 00:60:36:9f:c5:(21 + Node ID).

### Start CDI2 Answer 0x25

This just confirms reception of the Start CDI2 command. From this point on, the application must be able to receive and react to unsolicited messages from the BSM (e.g., BSM Status messages).

Offset	Length	Code/Data	Description
0	1	0x25	ID byte

### DET Status 0x16

Usually, the application issues this command to set the CDI2 device status in reaction to an BSM Status message. However, the application can decide to enter the error state, DS\_ERROR, at any point in the protocol.

Offset	Length	Code/Data	Description
0	1	0x16	ID byte
1	1		CDI2 Device State (Table 9)

### BSM Status 0x26

The BSM broadcasts its state in the synchronous phase of the DMB as part of the BSMINFO [Ref 1.]. Whenever the CDI2 IP Core detects a BSM status change, the application is informed with this unsolicited message.

Offset	Length	Code/Data	Description
0	1	0x26	ID byte
1	1		BSM State (Table 9)

#### 4.1.4. Feed-Off Phase

The BSM entering the FEED-OFF state gives the application an opportunity to perform an intensive self-test. As result of the self-test, the application can update the CDI2 maintenance state.

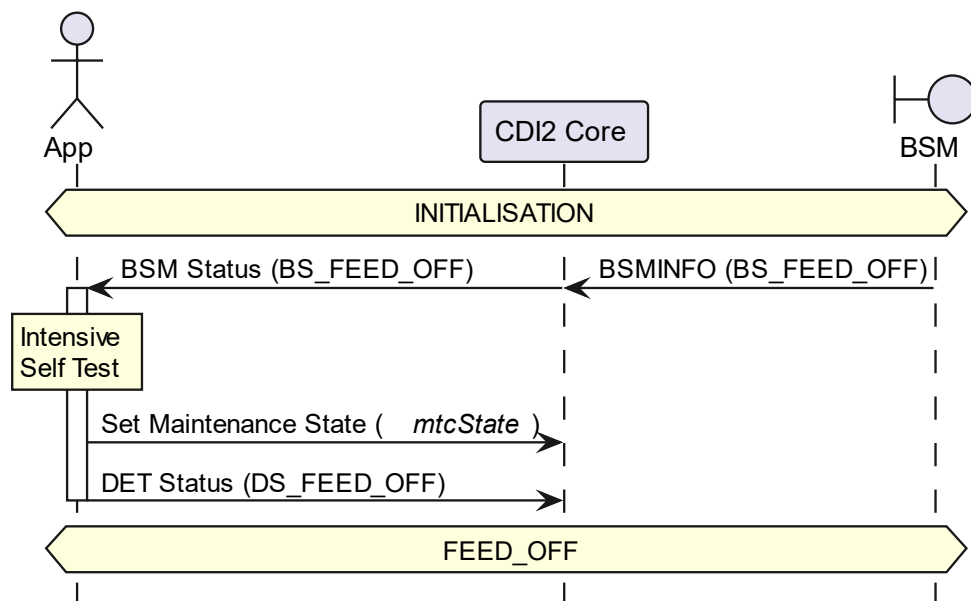


Figure 6: CDI2 Feed-Off

#### Set Maintenance State 0x17

Update the CDI2 maintenance state communicated to the BSM. Reserved bits should be set to 0. The current state can be queried at any time with the Protocol Version/Protocol Version Answer handshake.

Offset	Length	Code/Data	Description
0	1	0x17	ID byte
1	1		CDI2 Maintenance State [Ref 1.] Bit [0] ... air pulse required Bit [1] ... manual cleaning required Bit [2] ... self-test required Bit [3] ... calibration required Bit [7:4] ... reserved

## Set Error State 0x18

Update the CDI2 error state communicated to the BSM. Reserved bits should be set to 0. The current state can be queried at any time with the Protocol Version/Protocol Version Answer handshake.

Offset	Length	Code/Data	Description
0	1	0x18	ID byte
1	1		CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved

### 4.1.5. Sorting Phase

The BSM entering the REQUEST\_TO\_SORT state gives the application an opportunity to perform a short self-test. The BSM announces arriving banknotes at least 50mm ahead of the casing position [Ref 1.]. The CDI2 IP Core passes this information to the application with an unsolicited Banknote ID message that includes the Banknote ID allocated by the BSM and the remaining time it takes the BN to pass the casing position. The actual banknote arrival is then indicated by a Banknote Trigger message. Alternatively, the application can use the TTS\_BP (Table 1) signal as banknote trigger and ignore the trigger message.

Banknote information (series, denomination, and orientation) is conveyed with an unsolicited Banknote Info message to the application when it becomes available from the BSM.

The application sends its results with a Banknote Result command.

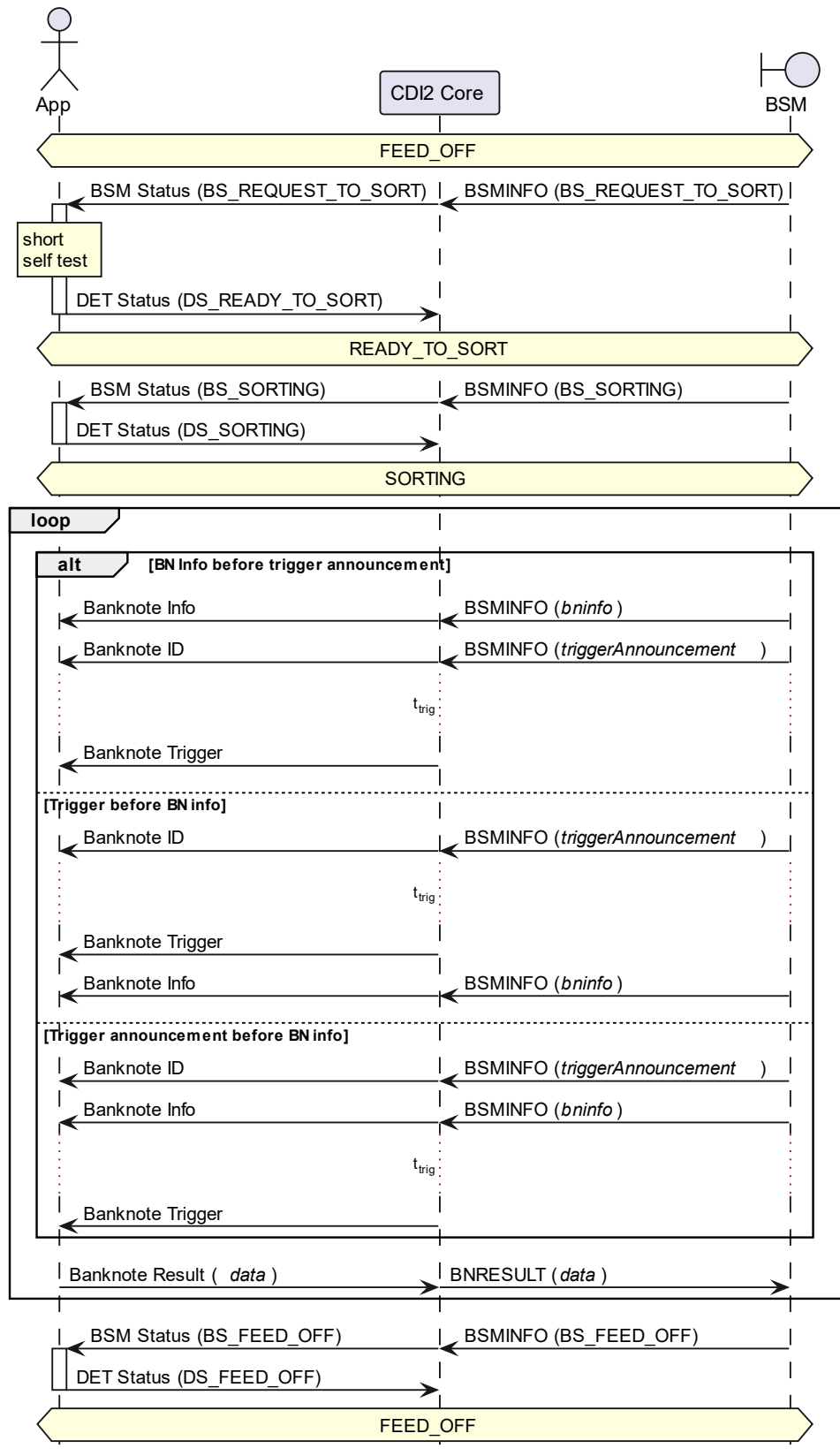


Figure 7: CDI2 Sorting



If the application acts as camera, an early recognition can be sent with the Banknote Recognition command

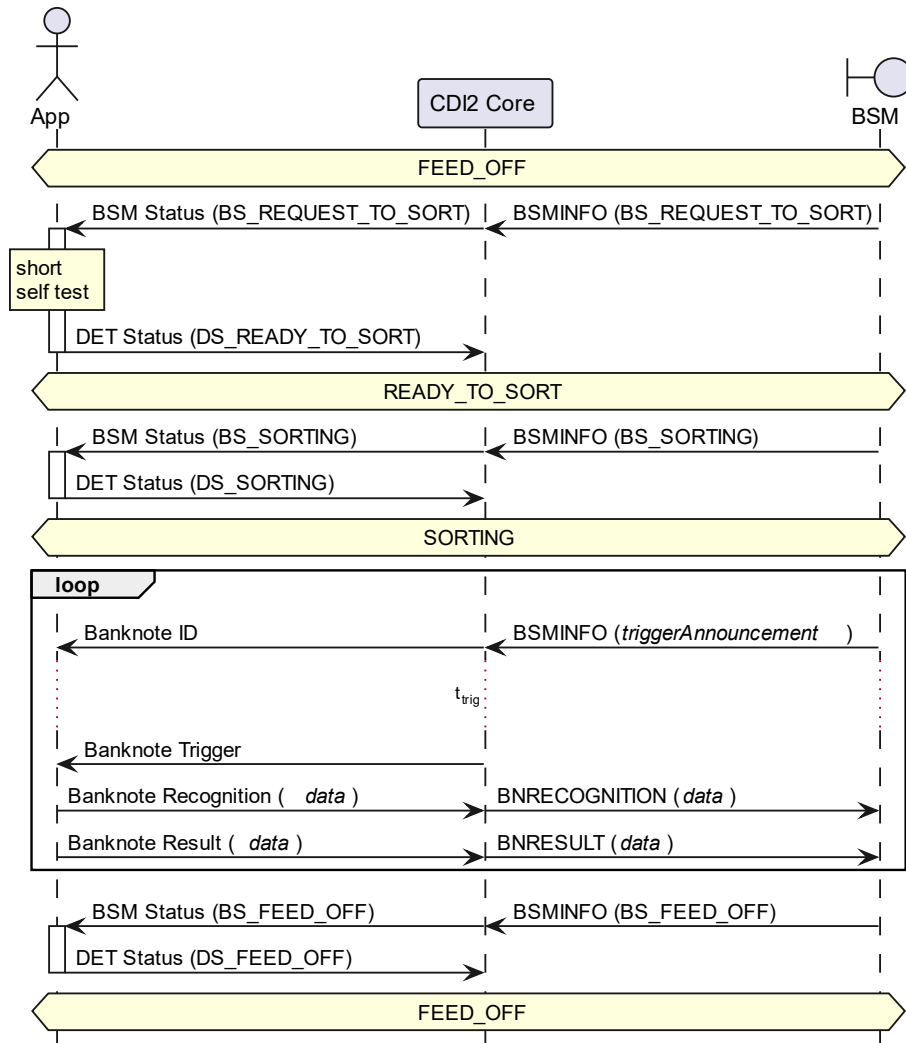


Figure 8: CDI2 Sorting (Camera)

## Banknote ID 0x81

The BSM allocates a banknote ID, an unsigned 32-bit integer, for every transported banknote. This message also contains trigger information that allows to application to determine the time instant the banknote will pass the casing position.

Offset	Length	Code/Data	Description
0	1	0x81	ID byte
1	4		Banknote ID
5	4		TC Count, the current transport clock counter, an unsigned 32-bit integer
9	4		TC Trigger, the trigger instant in TC clock counts, an unsigned 32-bit integer
13	4		$t_{\text{trig}}$ , the calculated arrival time [ns], an unsigned 32-bit integer

The CDI2 IP Core compensates the calculated arrival time by a nominal processing and transmission time, such that  $t_{\text{trig}}$  is measured from the 1<sup>st</sup> transmitted start bit of the PDU

### Banknote Trigger 0x80

If the application cannot use the TTS\_BP signal for a Banknote trigger, it can use the Banknote Trigger message from the CDI2 IP core.

Offset	Length	Code/Data	Description
0	1	0x80	ID byte
1	1	0 1	Trigger for BFA#0 Trigger for BFA#1
2	4		Banknote ID

### Banknote Info 0x82

After receiving a banknote recognition from a camera device, the BSM broadcasts the information in the BSMINFO section of the PRes. The CDI2 IP Core forwards this information to the application with this unsolicited message.

Offset	Length	Code/Data	Description
0	1	0x82	ID byte
1	4		Banknote ID
5	1		Banknote Series Codes [Ref 1.]
6	1		Banknote Denomination Codes [Ref 1.]
7	1		Banknote Orientation Codes [Ref 1.]

### Banknote Recognition 0x42

If acting as camera, a detector informs the BSM about series, denomination, and orientation of a processed banknote with this command. The CDI2 IP Core will queue this information to be put into the BNRECOGNITION section of the scheduled PRes frames on the DMB [Ref 1.].

Offset	Length	Code/Data	Description
0	1	0x42	ID byte
1	4		Banknote ID
5	1		Banknote Series Codes [Ref 1.]
6	1		Banknote Denomination Codes [Ref 1.]
7	1		Banknote Orientation Codes [Ref 1.]

### Banknote Result 0x41

The application informs the BSM about its sorting decision with the Banknote Result command [Ref 1.]. The CDI2 IP Core will queue this information to be put into the BNRESULT section of the scheduled PRes frames on the DMB [Ref 1.].

Offset	Length	Code/Data	Description
0	1	0x41	ID byte
1	4		Banknote ID
5	1		Banknote Series Codes [Ref 1.]
6	1		Banknote Denomination Codes [Ref 1.]
7	1		Banknote Orientation Codes [Ref 1.]
8	1	0x00	Reserved
9	1		Banknote Result Segment Number [Ref 1.]
10	3	0x000000	Reserved
13	N		Supplemental data, including judgment and result codes [Ref 1.]

### 4.1.6. Shutdown Phase

Entering the READY\_TO\_SHUTDOWN state is implemented with an otherwise unremarkable BSM Status message, DET Status command exchange. The CDI2 IP Core recognizes the BSM transition to BS\_SHUTDOWN in one of two ways:

- The BSM can broadcast its state BS\_SHUTDOWN before issuing the Stop Node NMT command
- Only the Stop Node command is received by the core

Regardless, the CDI2 IP Core informs the application about the BSM entering the shutdown state with the BSM Status message. However, in this state the core still monitors the TTS lines, so the BSM can restart the detector application via a TTS reset signalling.

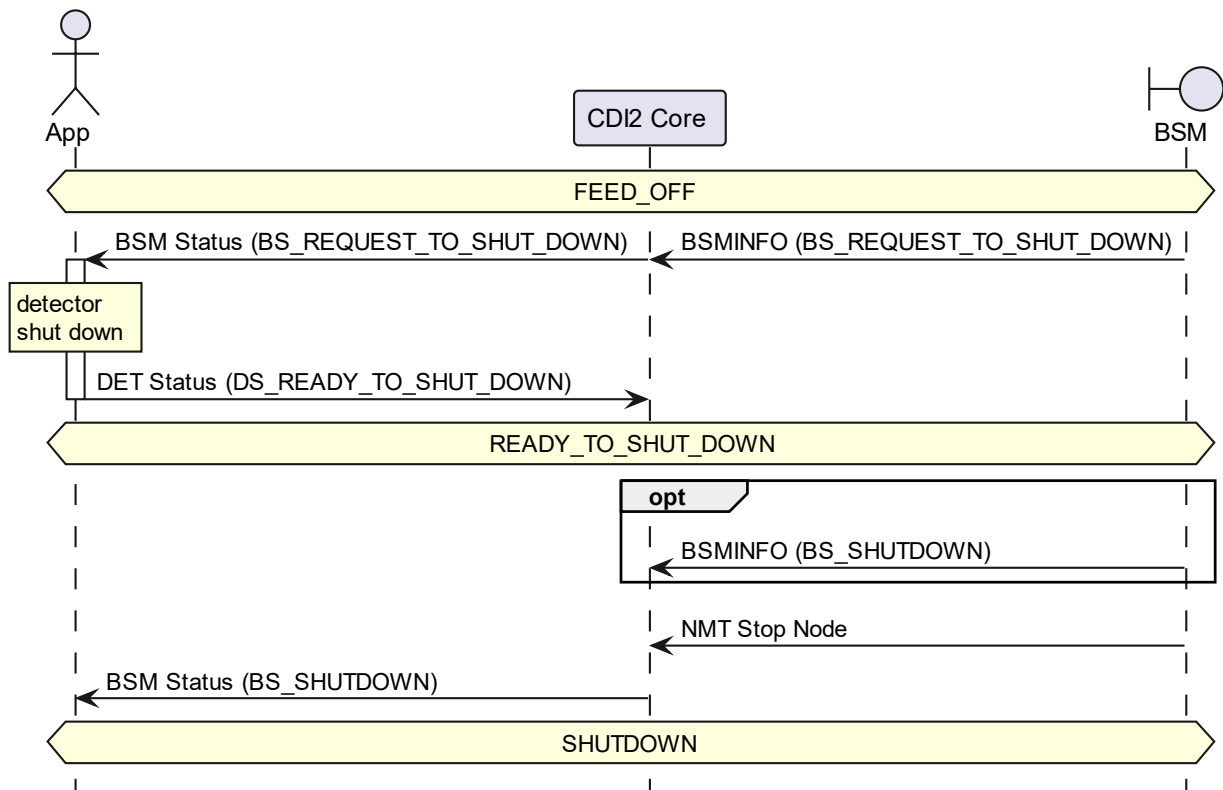


Figure 9: CDI2 Shut Down

#### 4.1.7. Error Handling and Logging

Next to the CDI2 device state, a CDI2 device also provides error and maintenance states in the PRes/BNRESULT Powerlink frames. The CDI2 IP Core provides commands, Set Maintenance State and Set Error State, to change these states. The application can query all current CDI2 and NMT states using the Protocol Version/Protocol Version Answer handshake. The CDI2 IP Core reports all errors, including Powerlink errors, with unsolicited FATAL or ERROR messages, depending on perceived severity. Cunningly, the error class corresponds to the definition of the CDI2 error state [Ref 1.]. Apart from FATAL and ERROR, the CDI2 IP Core supports with increasing verbosity, WARNING, INFO, DEBUG, and TRACE log messages. The verbosity of the CDI2 IP Core is set as parameter of the Start CDI2 command.

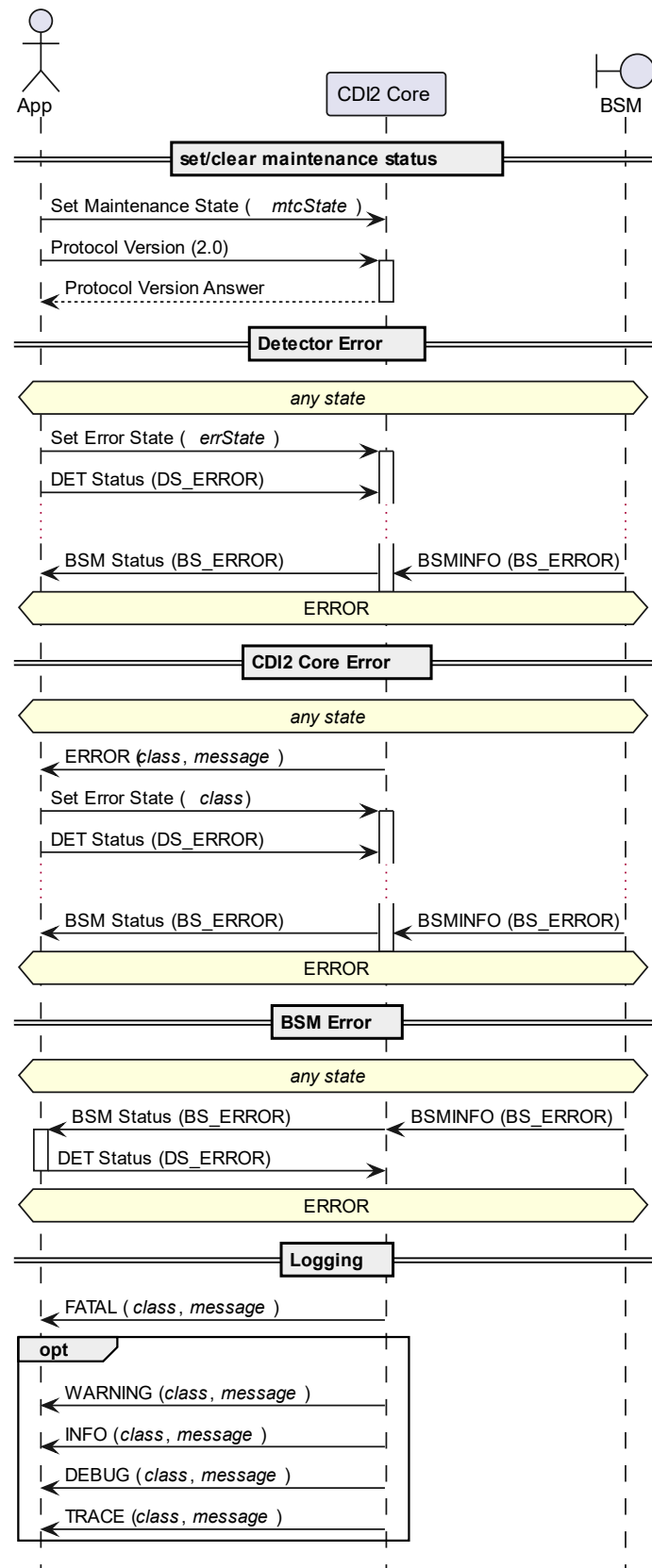


Figure 10: Error Handling and Logging

## FATAL 0x29

Reserved for severe error conditions that do not allow the CDI2 IP Core to proceed. Communication with the BSM cannot be assumed after this error.

Offset	Length	Code/Data	Description
0	1	0x29	ID byte
1	1		Error Classification/CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved
2	N		ASCIZ string describing the error

## ERROR 0x2A

Errors reported with this severity can usually be communicated to the BSM with a status transition to DS\_ERROR.

Offset	Length	Code/Data	Description
0	1	0x2A	ID byte
1	1		Error Classification/CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved
2	N		ASCIZ string describing the error

## WARNING 0x2B

This message reports unexpected events/behaviour or other potential problems that should not cause an immediate status transition to DS\_ERROR. This level can be activated to investigate and debug concrete problems.

Offset	Length	Code/Data	Description
0	1	0x2B	ID byte
1	1		Message Classification/CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved
2	N		ASCIZ string

### INFO 0x2C, DEBUG 0x2D, TRACE 0x2E

These messages report CDI2 IP Core progress, status, debug information and interface event traces of increasing verbosity. Sorting at full speed is not guaranteed with these messages enabled.

Offset	Length	Code/Data	Description
0	1	0x2C 0x2D 0x2E	ID byte INFO ID byte DEBUG ID byte TRACE
1	1		Message Classification/CDI2 Error State [Ref 1.] Bit [0] ... PowerLink Error Bit [1] ... TTS Error Bit [2] ... IDB Error Bit [3] ... Application Error Bit [7:4] ... reserved
2	N		ASCIZ string

#### 4.1.8. SW Update

The BSM transfers the new device SW as standard file, firmware.bin, with FTP to the application, then controls the update process with SDO write and read access to swUpdateCommand and swUpdateStatus. The CDI2 IP Core indicates changes of swUpdateStatus by the BSM to the application with Prepare Update and Perform Update messages. The application can react accordingly with Accept Update or Reject Update messages.

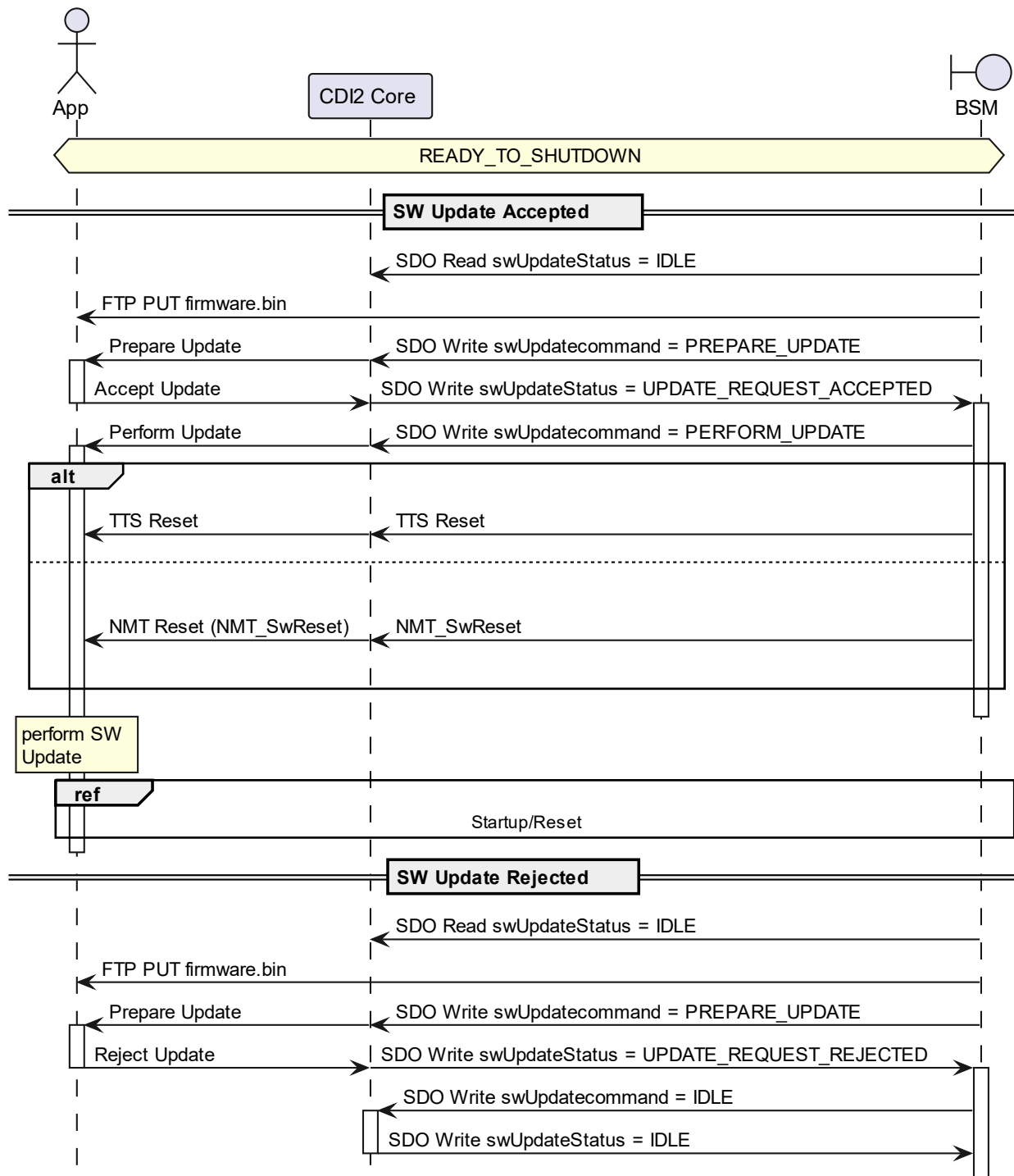


Figure 11: SW Update

### Prepare Update 0x87

After having transferred the new detector FW, firmware.bin, with FTP, the BSM asks the detector to check and prepare the update. The CDI2 IP Core indicates this BSM request to the application with this message.



Offset	Length	Code/Data	Description
0	1	0x87	ID byte

### Accept Update 0x47, Reject Update 0x48

After receiving the Prepare Update message, the application checks the new SW version and its general readiness to update the detector FW. If update is possible, it issues the Accept Update command, otherwise the Reject Update command. The BSM will eventually start the SW update proper with a Perform Update message if the update is accepted.

Offset	Length	Code/Data	Description
0	1	0x47 0x48	ID byte Accept Update ID byte Reject Update

### Perform Update 0x88

This message indicates to the application that the actual SW update should now commence.

Offset	Length	Code/Data	Description
0	1	0x88	ID byte

## 4.2. PPP:UART

While a full TCP/IP stack is not necessary for implementing the IP/DMB bridge function, providing the LCP [Ref 5.] and IPCP [Ref 6.] protocols in HDLC-like framing [Ref 14.] over the serial link allows the detector application using a standard PPP network interface for the FTP and HTTP/S servers on its side of the link. A minimal implementation of ARP [Ref 15.] is also necessary to learn the MAC addresses of MN/BSM and standard gateway. Both RTOS candidates, FreeRTOS [Ref 7.] and Quantum Leaps [Ref 8.], come with LwIP [Ref 9.], that provides the necessary protocol implementations. However, due to the tight memory and processing resources on the Nios/Microblaze processor, full implementation of all negotiable options is not feasible and will be denied. Specifically, for LCP: MRU = 1550, no ACFC, no PFC, no authentication, and no quality protocol. For IPCP: no negotiated IP address, no IP header compression, and no Van Jacobson TCP/IP header compression. Thus, a typical PPP session, as recorded by the Linux pppd(8) daemon, would look like:

```
Using interface ppp0
Connect: ppp0 <--> /dev/pts/2
sent [LCP ConfReq id=0x1]
rcvd [LCP ConfAck id=0x1]
sent [LCP ConfReq id=0x1]
rcvd [LCP ConfAck id=0x1]
rcvd [LCP ConfReq id=0x0]
sent [LCP ConfAck id=0x0]
kernel does not support PPP filtering
sent [IPCP ConfReq id=0x1 <addr 192.168.100.2>]
rcvd [IPCP ConfAck id=0x1 <addr 192.168.100.2>]
sent [IPCP ConfReq id=0x1 <addr 192.168.100.2>]
rcvd [IPCP ConfAck id=0x1 <addr 192.168.100.2>]
```

```
rcvd [IPCP ConfReq id=0x0 <addr 192.168.100.254>]
sent [IPCP ConfAck id=0x0 <addr 192.168.100.254>]
local IP address 192.168.100.2
remote IP address 192.168.100.254
Script /etc/ppp/ip-up started (pid 578)
Script /etc/ppp/ip-up finished (pid 578), status = 0x0
```

### 4.3. Test Concept

A separate test driver application will support the system test of the CDI2 IP Core.

For Xilinx targets the test platform will be an Enclustra ST3 base board [Ref 12.] with a Mars ZX2 SoC module [Ref 13.]. An AIT developed PHY board is used for the electrical DMB and TTS interfaces. The CDI2 IP Core will be instantiated in the FPGA fabric part of the SoC, while the test driver will execute on redundant PS as standard userland Linux application.



**Figure 12: Enclustra ST3 with a Mars SoC Module and AIT PHY board**

The test design will export the UART interface signals to external pins on the PHY board to allow an external test application to connect to the CDI2 IP Core instead of the test driver.

For Intel/Altera targets, the existing DMB simulator boards provide a convenient platform, because all electrical interfaces (DMB and TTS) are already present. The CDI2 IP Core will be instantiated in the FPGA fabric part of the SoC, while the test driver will execute on the HPS as standard userland Linux application.

The System test will be performed with the available CDI2 BSM simulator and test cases.

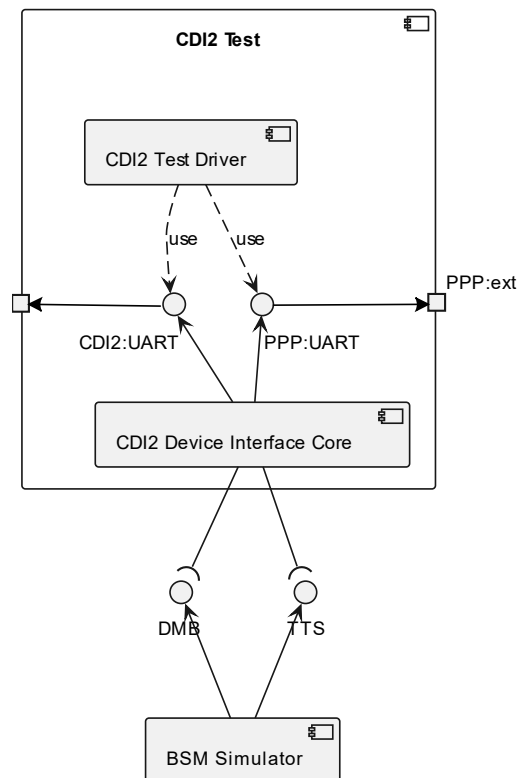


Figure 13: CDI2 IP Core Test

## 5. License Note

Any intellectual property contained in this document is presumably protected. This document does not grant a license for the use of intellectual property of AIT or third parties.

© Copyright

This document is copyrighted material of AIT, Vienna. All rights reserved. Any reproduction, publication or reprint in form of a different publication, in whole or in part, whether printed or produced electronically, is permitted only with the explicit prior written authorization of AIT.