Customer Training Workshop
Traveo™ II Serial Communication Block (SCB)
## Target Products

### Target product list for this training material

<table>
<thead>
<tr>
<th>Family Category</th>
<th>Series</th>
<th>Code Flash Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traveo™ II Automotive Body Controller Entry</td>
<td>CYT2B6</td>
<td>Up to 576KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2B7</td>
<td>Up to 1088KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2B9</td>
<td>Up to 2112KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2BL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller High</td>
<td>CYT3BB/4BB</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller High</td>
<td>CYT4BF</td>
<td>Up to 8384KB</td>
</tr>
<tr>
<td>Traveo II Automotive Cluster</td>
<td>CYT3DL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Cluster</td>
<td>CYT4DN</td>
<td>Up to 6336KB</td>
</tr>
</tbody>
</table>
Introduction to Traveo II Body Controller Entry

SCB is part of the Peripheral blocks
Introduction to Traveo II Body Controller High

SCB is part of the Peripheral blocks

---

**CPU Subsystem**

- Arm Cortex-M7
  - 350 MHz

**Peripheral Interconnect (MMIO, PPU)**

- eCT FLASH: 8384 KB Code flash + 256 KB Work flash
- SRAM0: 512 KB
- SRAM1: 256 KB
- SRAM2: 256 KB
- PMOS
- MDMAC
- MDMAC
- CRYPTO: AES, SHA, CRC, TRNG, RSA, ECC
- FLEXRAY Interface
- SWJ/ETM/ITM/CTI
- NVIC, MPU, AXI

**System Resources**

- Power
  - Sleep Control
  - POR
  - BOD
  - OVP
  - LVD
  - REF
  - PWRSYS-HT
  - LDO
- Clock
  - Clock Control
  - FLL
  - 4xPLL
- Reset
  - Reset Control
  - XRES
- Test
  - TestMode Entry
  - Digital DFT
  - Analog DFT
- WCO
- RTC

**Power Modes**

- Active/Sleep
- LowPower/Active/Sleep
- DeepSleep
- Hibernate

**I/O Subsystem**

- Up to 196x GPIO_STD, 4x GPIO_ENH, 40xHSIO
- High-Speed I/O Matrix, Smart I/O, Boundary Scan

---

Copyright © Infineon Technologies AG 2020. All rights reserved.

002-22210 *D  2020-12-28
Introduction to Traveo II Cluster

SCB is part of the Peripheral blocks

CPU Subsystem
- eCT FLASH 6336 KB Code flash + 128 KB Work flash

GFX Subsystem
- VRAM 4096 KB

Peripheral Interconnect (MMIO, PPU)
- IOSS GPIO
- PCLK
- 52x GPIO_STD, 8x GPIO_ENH, 26x GPIO_SMC, 70x HSIO_STD, 22x HSIO_ENH, 4x HSIO_ENG_DIFF

System Resources
- Power
  - Clock
  - Reset
  - Power Modes
- Clock Control
- PLL
- LDO
- WCO
- Power
- Active/Sleep
- DeepSleep

System Interconnect (Multi Layer AXI/AHB, IPC, MPU/SMPU)

Review TRM chapter 23 for additional details
Serial Communication Block Overview (1/3)

› Supports three serial interface protocols
  - Serial Peripheral Interface (SPI)
  - Universal Asynchronous Receiver/Transmitter (UART)
  - Inter-Integrated Circuit (I²C)

› SPI features
  - Supports master and slave functionality
  - Supports three SPI protocols (Motorola, Texas Instruments, National Semiconductor)
  - Supports up to four slave select lines
  - Programmable data frame size from 4 bits to 32 bits
  - Supports easy SPI (EZSPI) mode of operation
Serial Communication Block Overview (2/3)

UART Features

- Supports two protocols
  - Standard UART
  - Multi-processor mode
- Programmable data frame size from 4 to 16 bits
- Programmable number of STOP bits (can set half-bit periods between 1 and 4)
- Programmable oversampling

Hint Bar

Review TRM section 24.1 for additional details

Another dedicated LIN block can be used in Traveo II because this SCB has only standard LIN slave functionality
Serial Communication Block Overview (3/3)

I²C Features

- Master and slave mode
- Four types of data-rate modes
  - Slow mode, Standard mode, Fast mode, Fast mode plus
- 7-bit slave addressing
- Analog and digital glitch filter – up to 50 ns

Hint Bar
Review TRM section 1 for additional details
Serial Communication Block Diagram

Block Diagram

Review TRM section 2 for additional details
Serial Communication Block Components

- SPI
  - Supports up to four slave select lines
  - Supports three SPI protocols
  - SPI Buffer mode
    - FIFO mode
    - EZSPI mode
    - Command-Response mode

Hint Bar

Review TRM section 24.4 for additional details
The standard SPI interface consists of four signals:
- SCLK: Serial clock (clock output from the master, input to the slave)
- MOSI: Master-out-slave-in (data output from the master, input to the slave)
- MISO: Master-in-slave-out (data input to the master, output from the slave)
- Slave Select (SS): Typically an active-low signal (output from the master, input to the slave)

Advantage:
- Can communicate with multiple external ICs using the same bus
SPI Protocols

- Three SPI protocols\(^1\)
  - The Motorola SPI\(^2\) protocol has four clock modes
    - Data is driven and captured by clock polarity and clock phase
  - Texas Instruments SPI\(^2\)
    - The Texas Instruments SPI protocol redefines the use of the SS signal to indicate the start of a data transfer
  - National Semiconductor SPI\(^3\)
    - The National Semiconductor SPI protocol alternates transmission and reception

---

\(^1\) Refer to the Register TRM (SCBx_SPI_CTRL) for additional details
\(^2\) Full Duplex
\(^3\) Half Duplex
Motorola SPI Mode

- The Motorola SPI protocol has four clock modes based on how data is driven and captured on the MOSI and MISO lines.
- The modes are determined by clock polarity (CPOL) and clock phase (CPHA).

<table>
<thead>
<tr>
<th>Mode</th>
<th>CPOL</th>
<th>CPHA</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Data is driven on a falling edge of SCLK. Data is captured on a rising edge of SCLK.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Data is driven on a rising edge of SCLK. Data is captured on a falling edge of SCLK.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Data is driven on a rising edge of SCLK. Data is captured on a falling edge of SCLK.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Data is driven on a falling edge of SCLK. Data is captured on a rising edge of SCLK.</td>
</tr>
</tbody>
</table>

- Motorola SPI data transfer example (Mode = 0)
  - A single 8-bit data transfer

Legend:
- CPOL: Clock Polarity
- CPHA: Clock Phase
- SCLK: SPI Interface Clock
- MOSI: SPI Master-Out-Slave-In
- MISO: SPI Master-In-Slave-Out

1 Refer to the Register TRM (SCBx_TX_CTRL, SCBx_RX_CTRL) for additional details.
Texas Instruments SPI Mode

- Texas Instruments SPI data transfer
  - A single 8-bit data transfer
  - The Slave Select pulse either precedes the first data bit or coincides with the first data bit of a frame
    - Slave Select pulse preceding example
    - Slave Select pulse coinciding example

Legend:
- CPOL: Clock Polarity
- CPHA: Clock Phase
- SCLK: SPI Interface Clock
- MOSI: SPI Master-Out-Slave-In
- MISO: SPI Master-In-Slave-Out

1 Texas Instruments SPI mode uses only CPOL = 0 and CPHA = 1 settings
National Semiconductor SPI Mode

- National Semiconductor SPI data transfer
  - The SPI protocol alternates transmission and reception
  - The transmission and reception data sizes are different
  - A single “idle” bit transfer period separates transmission from reception
SPI Buffer Modes (1/2)

- SPI can operate in three buffer modes:
  - FIFO mode
  - EZSPI mode
  - Command-Response mode
- Restrictions on using buffer mode:
  - Master can only use FIFO mode
  - Slave can use all above three buffer modes. However, Command-Response mode is available only on DeepSleep-capable SCB
- SCB has an internal SRAM buffer:
  - Capacity of $128 \times 16$ bits
  - Supports byte mode
  - Can be configured as $256 \times 8$ bits

1 Refer to the Register TRM (SCBx_CTRL.MODE) for additional details
Buffer mode can be configured in five types in dedicated SRAM:

1. Two 32 deep FIFOs for up to 32-bit data elements
2. Two 64 deep FIFOs for up to 16-bit data elements
3. Two 128 deep FIFOs for up to 8-bit data elements
4. One 256 Byte EZ memory buffer
5. One 256 Byte CMD_RESP memory buffer

Refer to the Register TRM (SCBx_CTRL.EZ_MODE, SCBx_CTRL.CMD_RESP_MODE, SCBx_CTRL.MEM_WIDTH) for additional details.
FIFO Mode

- FIFO mode includes TX and RX FIFOs
- Transmit and receive FIFO status\(^1\) information indicates
  - Empty
  - Not empty
  - Full
  - Not full
- Provides “underflow” and “overflow” events
  - An “underflow” event is triggered by an attempt to read from an empty FIFO
  - An “overflow” event is triggered by an attempt to write to a full FIFO

\(^1\) Refer to the Register TRM (SCBx_INTR_TX, SCBx_INTR_RX) for additional details
EZSPI Mode

The EZSPI\(^1\) protocol is based on the Motorola SPI
- The EZSPI mode is a method in which the master uses an 8-bit EZ address in the slave memory to write or read data
- It defines a single memory buffer with an 8-bit EZ address
- The EZ address indexes the buffer located on the slave device
- EZSPI has three types of transfers
  - EZ address written from the master to the slave
  - Data written from the master to an addressed slave memory location
  - Slave reads from an addressed slave memory location

\(^1\) Refer to the Register TRM (SCBx_CTRL.EZ_MODE) for additional details
EZSPI Mode (Address Write)

EZ Address Write

- An EZ address write starts with a command byte (0x00) on the MOSI line indicating the master’s intent to write the EZ address

Hint Bar
Review TRM section 4.4.2 for additional details
EZSPI Mode (Data Write)

- Memory Array Write
  - A write to a memory array index starts with a command byte (0x01) on the MOSI line indicating the master’s intent to write to the memory array
  - The EZ address is automatically incremented by the slave as bytes are written into the memory array

Command 0x01: Write Data

![Diagram showing SPI signals and command sequence]

**Legend:**
- CPOL: Clock Polarity
- CPHA: Clock Phase
- SCLK: SPI Interface Clock
- MISO: SPI Master-In-Slave-Out
- MOSI: SPI Master-Out-Slave-In
  - 0x00: Write EZ address
  - 0x01: Write DATA
  - 0x02: Read DATA
  - 0xFE: "slave ready"
  - 0xFF: "slave busy"

**Hint Bar:**
Review TRM section 4.4.2 for additional details.
EZSPI Mode (Data Read)

› Memory Array Read
  - A read from a memory array index starts with a command byte (0x02) on the MOSI line indicating the master’s intent to read from the memory array

› Advantage
  - Effective transmission and reception without using software to check the memory buffer address

Hint Bar
Review TRM section 4.4.2 for additional details
Command-Response Mode

- Only supports an SPI slave
- This mode has a single memory buffer, which is indexed using
  - Base read/write address
  - Current read/write address
- The command-response mode has two phases of operation
  - Write phase
  - Read phase
- The slave transmits either 0x62 (ready) or another value (busy)
- When disabled or reset, the slave transmits 0xFF (busy)
Command-Response Mode (Write/Read)

- The command-response write/read phases are
  - **Write phase**
    - Write phase begins with a selection byte (last bit set to ‘0’ indicating a write)
    - Master writes 8-bit data elements to the slave’s memory buffer
      (the current write address is incremented)
  - **Read phase**
    - Read phase begins with a selection byte (last bit set to ‘1’ indicating a read)
    - Master reads 8-bit data elements from the slave’s memory buffer
      (the current read address is incremented)

- **Advantage**
  - Effective transmission and reception without using software to check the memory buffer address
## Interrupt Events in SPI Mode

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>SPI master transfer done</td>
</tr>
<tr>
<td>RX</td>
<td>SPI Bus Error</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is not full</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is empty</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO overflows</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO underflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is full</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is not empty</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO overflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO underflows</td>
</tr>
</tbody>
</table>

Hint Bar: Review TRM section 7.1 for additional details.
UART Component Serial Communications Block

- The UART in the SCB block supports the following:
  - Standard UART
  - Multi-processor mode
  - Start skipping function
  - Oversampling

Hint Bar

Review TRM section 24.5 for additional details
Standard UART

- TX and RX connection
- Buffer mode only uses FIFO mode

Typical UART transfer frame components
- Start bit
- Data bits (programmable from 4 to 16 bits)
- Parity bit (optional)
- Stop bit (can be set to half-bit periods between 1 and 4)

Hint Bar
Review TRM section 5.3.1 for additional details
Multi-Processor Mode

› UART_MP (Multi-Processor) mode
  - Single-master-multi-slave connection

› UART_MP mode properties
  - Each slave is identified by a unique address
  - Uses a 9-bit data field
    - Ninth bit as address/data flag (MP bit)
  - Parity bit is disabled

› Advantage
  - Can communicate with multiple external ICs using the same bus
Start Skipping Function

Use Start Skipping to wake up from DeepSleep mode

1. The GPIO is set as an interrupt to UART RX line transition ‘1’ to ‘0’ (START bit)
2. The UART receiver is not functional during DeepSleep mode transition
3. When the GPIO interrupt is activated, the system transits from DeepSleep to Active mode
4. The CPU enables UART receive functionality
5. The UART receiver synchronizes with the receipt data frame
6. The UART receiver proceeds with normal operation

Advantage
- UART can receive data immediately upon MCU wakeup from DeepSleep

Hint Bar
Review TRM section 5.3.1 for additional details
Oversampling

- The receiver oversamples the incoming signal
- The sampling clock can select from 8 to 16 clock cycles by software
- The value of the sample point in the middle of the bit transfer period is used for transmitter and receiver clock synchronization

- Advantage
  - Can be used to synchronize the receiver with the transmitter clock

1 Alternatively, three samples around the middle of the bit transfer period are used for increased accuracy
## Interrupt events in UART mode

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>TX done</td>
</tr>
<tr>
<td>RX</td>
<td>Frame error in received data frame</td>
</tr>
<tr>
<td>RX</td>
<td>Parity error in received data frame</td>
</tr>
<tr>
<td>RX</td>
<td>LIN baud rate detection is completed</td>
</tr>
<tr>
<td>RX</td>
<td>LIN break detection is successful</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is not full</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is empty</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO overflows</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO underflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is full</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is not empty</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO overflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO underflows</td>
</tr>
</tbody>
</table>
I²C Component in Serial Communications Block

- I²C supports the following:
  - Standard I²C
  - I²C Buffer mode
    - FIFO mode
  - EZI²C mode
  - Command-Response mode
  - Glitch filtering

1 FIFO mode has the same features as SPI
Standard I\textsuperscript{2}C

- The standard I\textsuperscript{2}C bus is a two-wire interface with these lines:
  - Serial Data (SDA)
  - Serial Clock (SCL)
- I\textsuperscript{2}C devices are connected to these lines through pull-up
- Masters and slaves can operate as either transmitter or receiver
Write Transfer

› The master sends data to the slave via a typical write transfer
  - Generate START condition
  - 7-bit \( I^2 C \) slave address and a write indicator ‘0’ bit
  - The addressed slave transmits an acknowledgment
  - If the master receives an acknowledgment, it transmits data (8 bits) to the bus
  - When the transfer is complete, the master generates a STOP condition

Legend:
- **SDA**: Serial Data Line
- **SCL**: Serial Clock Line (always driven by the master)
- **Slave Transmit / Master Receive**

Hint Bar
Review TRM section 6.4.1 for additional details
Read Transfer

- The master receives data from the slave via a typical read transfer
  - Generate START condition
  - 7-bit \( \text{I}^2\text{C} \) slave address and a read indicator ‘1’ bit
  - If the slave acknowledges the acknowledgment signal from the master, it starts transmitting data after the acknowledgment signal
  - The master transmits an acknowledgment to confirm
  - When the transfer is complete, the master generates a STOP condition

![Diagram of I2C bus data transfer](image)

**LEGEND:**
- SDA: Serial Data Line
- SCL: Serial Clock Line (always driven by the master)
- **Gray**: Slave Transmit / Master Receive

**Hint Bar:**
Review TRM section 6.4.2 for additional details
I²C Buffer Modes (1/2)

› I²C can operate in three buffer modes¹
  - FIFO mode²
  - EZI2C mode
  - Command-Response mode

› Restrictions on using buffer mode³
  - Master can only use FIFO mode
  - Slave can use all above three buffer modes. However, Command-Response mode is available only on DeepSleep-capable SCB

› SCB has an internal SRAM buffer³
  - Capacity of 128 x 16 bits
  - Supports byte mode
  - Can be configured as 256 x 8 bits

¹ Refer to the Register TRM (SCBx_CTRL.MODE) for additional details
² FIFO mode has the same features as SPI
³ These features are the same as the SPI Buffer mode
I²C Buffer Modes (2/2)

Buffer mode can be configured in five types in dedicated SRAM

1. Two 32 deep FIFOs for up to 32-bit data elements
2. Two 64 deep FIFOs for up to 16-bit data elements
3. Two 128 deep FIFOs for up to 8-bit data elements
4. One 256 Byte EZ memory buffer
5. One 256 Byte CMD_RESP memory buffer

1 Refer to the Register TRM (SCBx_CTRL_EZ_MODE, SCBx_CTRL_CMD_RESP_MODE, SCBx_CTRL_MEM_WIDTH) for additional details. These features are the same as the SPI Buffer mode.
EZI2C Mode

EZI2C distinguishes three operation phases
- Address phase\(^1\)
  - The master transmits an 8-bit EZ address to the slave
- Write phase\(^1\)
  - The master writes 8-bit data elements to the slave’s memory buffer
- Read phase\(^1\)
  - The master reads 8-bit data elements from the slave’s memory buffer

Advantage
- Effective transmission and reception without using software to check the memory buffer address

\(^1\) These operations are the same as the EZ SPI mode
Command-Response Mode

- Only supports an I²C slave
- This mode has a single memory buffer, which is indexed using\(^1\)
  - Base read/write address
  - Current read/write address
- The command-response mode has two phases of operation
  - Write phase
  - Read phase

\(^1\) The features are same as the SPI Command response mode
The command-response write/read phases are

- **Write phase**
  - Begins with a START bit, slave address, and write bit (set to ‘0’ indicating a write)
  - The master writes 8-bit data elements to the slave’s memory buffer (the current write address is incremented)

- **Read phase**
  - Begins with a START bit, slave address, and read bit (set to ‘1’ indicating a read)
  - The master reads 8-bit data elements from the slave’s memory buffer (the current read address is incremented)

**Advantage**
- Effective transmission and reception without using software to check command

1 The operation is the same as the SPI Command response mode
Glitch Filtering

- The Traveo II SCB I²C has analog and digital glitch filters
- Analog glitch filters (AF_in, AF_out) are applied in the following:
  - i2c_scl_in
  - i2c_sda_in
  - i2c_sda_out
- Analog glitch filters are applied to filter glitches of up to 50 ns
- Digital glitch filters (DF_in) are applied in the following:
  - i2c_scl_in
  - i2c_sda_in
- Digital glitch filters are 3-tap median filters

Hint Bar

Review TRM section 6.6.1 for additional details
## Interrupt

### Interrupt events in I²C mode

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>I²C master lost arbitration</td>
</tr>
<tr>
<td>RX</td>
<td>I²C master received NACK</td>
</tr>
<tr>
<td>RX</td>
<td>I²C master received ACK</td>
</tr>
<tr>
<td>TX</td>
<td>I²C master sent STOP</td>
</tr>
<tr>
<td>RX</td>
<td>I²C master bus error (unexpected stop/start condition detected)</td>
</tr>
<tr>
<td>TX</td>
<td>I²C slave lost arbitration</td>
</tr>
<tr>
<td>RX</td>
<td>I²C slave received NACK</td>
</tr>
<tr>
<td>RX</td>
<td>I²C slave received ACK</td>
</tr>
<tr>
<td>RX</td>
<td>I²C slave received STOP</td>
</tr>
<tr>
<td>RX</td>
<td>I²C slave received START</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX</td>
<td>I²C slave address matched</td>
</tr>
<tr>
<td>RX</td>
<td>I²C slave bus error (unexpected stop/start condition detected)</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is not full</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO is empty</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO overflows</td>
</tr>
<tr>
<td>TX</td>
<td>TX FIFO underflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is full</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO is not empty</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO overflows</td>
</tr>
<tr>
<td>RX</td>
<td>RX FIFO underflows</td>
</tr>
</tbody>
</table>

Review TRM section 7.3 for additional details
Part of your life. Part of tomorrow.
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>6178141</td>
<td>05/21/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6415893</td>
<td>12/10/2018</td>
<td>Added slides 2, 4, 5, 8, and the note descriptions on all pages. Updated slides 3 and 7.</td>
</tr>
<tr>
<td>*B</td>
<td>6668243</td>
<td>09/04/2019</td>
<td>Updated slides 2, 3, 4, 6, 7, 8, 11,18 and 21. Added slide 5.</td>
</tr>
<tr>
<td>*C</td>
<td>6702599</td>
<td>10/16/2019</td>
<td>Corrected spec footer revision.</td>
</tr>
<tr>
<td>*D</td>
<td>7062561</td>
<td>01/08/2021</td>
<td>Updated slides 2, 3, 9, 10, 12, 13, 16, 18, 19, 26, 27, 32 and 36. Added slide 17 and 37.</td>
</tr>
</tbody>
</table>