Customer Training Workshop

Traveo™ II Clock Extension Peripheral Interface (CXPI)
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>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 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

The CXPI controller is part of Peripheral Blocks

System Resources

- Power
  - PWRSYS-HT
  - LDO
- Clock
  - PLL
  - TCPLL
- Reset
  - RWST
  - TMR
  - HWB
  - SWRST
  - SWRST
- Analog
  - SAR
  - SARMUX
  - 64 ch
- Test
  - Test
- Power Modes
  - DeepSleep
  - LowPower/Active/Sleep
  - Hibernate

CPU Subsystem

- CPU Subsystem
  - Arm Cortex M4 160 MHz
  - FPB, NVIC, MPU
  - 8 KB $ FLASH Controller
  - 8 KB $ FLASH Controller
  - SRAM0 256 KB
  - SRAM1 256 KB
  - PMAM0
  - CRYPTO
    - AES, SHA, CRC, TRNG, RSA, ECC
    - Initiator/MMIO
  - SWMI/MMCT
  - Arm Cortex M0+ 100 MHz
  - ROM 32 KB

Peripheral Interconnect (MMIO, PPU)

- Peripheral Interconnect
- SWI/ETMI/MMCT
- eCT Flash
  - 4160 KB Code-flash + 128 KB Work-flash
  - SRAM Controller
  - SRAM Controller
  - 7x SCB
  - I2C, SPI, UART
  - I2C, SPI, UART
  - 1x SCB
  - 6x LIN
  - LIN/UART
  - CV TPWM
  - TIMER, CTR, QD, PWM
  - 83x TCPWM
  - 12x LIN
  - LIN/UART
  - 128 K flash + 128 K flash
  - 4160 KB Code
  - 8 KB $ $
  - 32 KB ROM
  - SRAM0 256 KB
  - SRAM1 256 KB
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller

System Interconnect (Multi Layer AHB, IPC, MPU/SMTP)

- System Interconnect
- SWI/ETMI/MMCT
- Arm Cortex M4 160 MHz
- FPB, NVIC, MPU
- 8 KB $ FLASH Controller
- 8 KB $ FLASH Controller
- SRAM0 256 KB
- SRAM1 256 KB
- PMAM0
- CRYPTO
  - AES, SHA, CRC, TRNG, RSA, ECC
  - Initiator/MMIO
- SWMI/MMCT
- Arm Cortex M0+ 100 MHz
- ROM 32 KB

I/O Subsystem

- I/O Subsystem
  - SWI/ETMI/MMCT
  - eCT Flash
    - 4160 KB Code-flash + 128 KB Work-flash
  - SRAM Controller
  - SRAM Controller
  - 7x SCB
  - I2C, SPI, UART
  - I2C, SPI, UART
  - 1x SCB
  - 6x LIN
  - LIN/UART
  - CV TPWM
  - TIMER, CTR, QD, PWM
  - 83x TCPWM
  - 12x LIN
  - LIN/UART
  - 128 K flash + 128 K flash
  - 4160 KB Code
  - 8 KB $ $
  - 32 KB ROM
  - SRAM0 256 KB
  - SRAM1 256 KB
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller
  - SRAM Controller

Power

- Power
  - SWMI/MMCT
  - Arm Cortex M4 160 MHz
  - FPB, NVIC, MPU
  - 8 KB $ FLASH Controller
  - 8 KB $ FLASH Controller
  - SRAM0 256 KB
  - SRAM1 256 KB
  - PMAM0
  - CRYPTO
    - AES, SHA, CRC, TRNG, RSA, ECC
    - Initiator/MMIO
  - SWMI/MMCT
  - Arm Cortex M0+ 100 MHz
  - ROM 32 KB

DeepSleep

- Hibernate

Power Modes

- Hibernate
- DeepSleep
- LowPower/Active/Sleep

Copyright © Infineon Technologies AG 2020. All rights reserved.
Introduction to Traveo II Cluster

The CXPI controller is part of Peripheral Blocks

- CYT4DN
- MXS40-HT
- ASIL-B

System Resources
- Power
  - Sleep Control
  - PWRSYS_H
  - LDO
- Clock
  - PLL
  - DV
  - PLL
  - LPECO
- Reset
  - Test
  - LPLockout
- Backend
  - Digital DFT
  - Analog DFT
- SRAM
  - SRAM1
  - 128 KB
  - SRAM2
  - 256 KB
- SRAM3
  - 64 KB
  - SRAM4
  - 128 KB
- SRAM5
  - 256 KB

CPU Subsystem
- Cortex M7
  - 320 MHz
- eCT Flash
  - 8336 KB Code-flash
- SRAM0
  - 256 KB
- SRAM1
  - 256 KB
- SRAM2
  - 128 KB
- System Interconnect
  - Multi Layer AXI/AHB, IPC, MPU/SMPU
  - High Speed I/O Matrix
  - Smart I/O
  - Boundary Scan

Peripheral Interconnect (MMIO, PPU)
- Prog. Analog
- SAR ADC
- 12-bit
- SARMUX
- 48 ch

GFX Subsystem
- Cortex M0+
  - 100 MHz
- ROM
  - 64 KB
- ROM Controller
- VRAM
  - 4096 KB
- GFX Interconnect (AXI)
- 2x SMIF
- 2x SCB
- I²C, SPI, UART, LIN
- 2x LIN
- 2x UART
- 2x RGB/MIPI Input
- 2x RGB/LVDS Output
- 4096 KB VRAM
- 64 KB ROM Controller

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

Copyright © Infineon Technologies AG 2020. All rights reserved.
CXPI Overview

› Clock Extension Peripheral Interface (CXPI) is a single-line communication bus with clock modulation to synchronize all slave nodes with the master clock
› Features
  - CXPI protocol support in hardware according to ISO/WD 20794-4
  - Master and Slave nodes
    - Autonomous request field and response transfer processing
  - Data signal encoding and decoding formats
    - Non-return to zero (NRZ) mode
    - Pulse-width modulation (PWM) mode
  - Wake pulse generation
  - Error detection
  - Timeout detection
  - Test modes including hardware error injection
CXPI Block Diagram

CXPI module block component

- From PERI/PCLK
- clk_peri
- PCLK_PERI
- Trigger to P-DMA
- CXPI_TR_tx_REQ[i]
- CXPI_TR_rx_REQ[i]
- cxpi_interrupts_[i]_IRQn
- TCPWM_TO_CXPI_TR[i]
- Trigger from TCPWM IP
- Definition:
  - i: channel
  - CH_NR: max. channel number
  - clk_peri
  - PCLK_PERI

From To IOSS/HSIOM

FIFO

CXPI I/O Signal Router

CXPI Channel [i]

AHB Slave IF

Test Registers

CXPI Controller

CXPI Module Block

Review TRM section 30.2 for additional details
CXPI Module Block Components

- CXPI Channel
- Message Frame Format
- Message Transfer Processing Commands
- Message Frame Transfer
- Arbitration Loss
- Bus Signal Modulation
- FIFO Buffer
- P-DMA Trigger
- Timeout Detection
- Enabling CXPI Channel
- Power Modes
- Oversampling
- Noise filter
Message Frame Format (1/2)

Normal Frame

Protected Identifier (PID) field:
- 7-bit frame identifier
- 1-bit odd parity over frame identifier

The DLC value signifies data bytes only and does not include the CRC byte.

Protected Type (PTYPE) field:
- 8-bit of PTYPE
- Master sends PTYPE byte to permit all slave nodes to send a request field for this time slot
- Only applicable in Polling method

Frame Information (FI) field:
- 2 bits Counter (CT)
- 2 bits Network Management (NM)
- 4 bits Data Length Code (DLC) (Up to 12 bytes)

Inter Byte Space (IBS): The idle time between two bytes within a message frame

CRC field: 8 bits

Protected Identifier (PID) field:
- 7-bit frame identifier
- 1-bit odd parity over frame identifier

The DLC value signifies data bytes only and does not include the CRC byte.
Message Frame Format (2/2)

› Long Frame

![Diagram showing message frame format]

- Header/PTYPE
- Header/PID
- Message frame
- Response

CXPI bus

- Frame ID
- Parity bit
- PID bits
- Frame Information bits

PTYPE - PID - FI - DLC Ext - Data 1 - Data 2 - N - CRC - CRC

Data N

- Data Length Code Extension (DLCEXT) field:
  - 8 bits of DLCEXT (Up to 255 bytes)

CRC field: 16 bits

1 The DLCEXT value signifies data bytes only and does not include the CRC byte.
Message Transfer Processing Commands

- These commands are set in the CMD register

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_HEADER</td>
<td>Transmit a header</td>
</tr>
<tr>
<td>TX_RESPONSE</td>
<td>Transmit a response</td>
</tr>
<tr>
<td>IFS_WAIT</td>
<td>Check if the bus is in idle state before transmitting</td>
</tr>
<tr>
<td>RX_HEADER</td>
<td>Receive a header</td>
</tr>
<tr>
<td>RX_RESPONSE</td>
<td>Receive a response</td>
</tr>
</tbody>
</table>

- Advantage
  - Reduces CPU load
Message Frame Transfer

› Method 1: Event Trigger Method
  – The master and each slave node can start a frame when the bus is ready for transmission
  – Advantage:
    – High responsivity to the event

› Method 2: Polling Method
  – Message transfers are scheduled in general and thereby every transfer is initiated by the master
  – Advantage:
    – Ensures the periodicity of the communication

Review TRM section 30.5.1 for additional details
Either method can be chosen and implemented according to the system requirements
Method 1: Event Trigger Method Example

1. Node A transmits Request ID B after detecting that the bus is in idle state
2. Node B, corresponding to Request ID B, transmits Response B
3. Event occurs on node B and node C
4. Request ID B and Request ID C are transmitted at the same time when the bus is in idle state
5. Collision occurs
6. Request ID C (node C) is transmitted as a result of arbitration and Response C is retransmitted
7. Node B transmits Request ID B based on the time when the bus moves to idle state

Arbitration takes place according to PID field priority. The smaller frame ID with the bit of logical value ‘0’ has priority over the bit of logical value ‘1’.
Method 2: Polling Method Example

1. Node A transmits Request ID B after detecting that the bus is in idle state
2. Node B, corresponding to Request ID B, transmits Response B
3. Event occurs on node B1 and node C
4. Node A sends a PTYPE field to permit all slave nodes to send a request field for this time slot
5. Request ID B and Request ID C are transmitted at the same time
6. Collision occurs
7. Request ID C (node C) is transmitted as a result of arbitration and Response C is retransmitted
8. Node C transmits response so that node A transmits frame ID C of the node C
Arbitration Loss

- Defined as TX and RX bit mismatch during transmission of the PID field or PTYPE field\(^1\)
- If an arbitration is lost, HW checks whether it has reached its maximum amount of retries via CTL2.RETRY

\(^{1}\) Excluding Start bit or Stop bit
Event Trigger Method Operation (1/2)

Transmission of PID and RESPONSE by Master/Slave

Step | Software Processing | CXPI Hardware Processing
--- | --- | ---
(0) | 1. Channel initialization:
- Set master/slave and modulation mode (NRZ/PWM)
- Automatic Transceiver Handling ON/OFF
- Receive PID Zero Check OFF (only for slave in Polling method)
- RX filtering ON/OFF
- Set IFS length and IBS length
- TX abort for bit error detection ON/OFF
- Define PWM pulses (only PWM mode)
- Configure RX sample point
- Define TX wake-up pulse length
- Select Time-out and configure FIFO
2. Start:
- Enable Channel and transit CXPI bus to Normal mode | 1. Channel disabled

(1) | - PID field Transmission (master/slave):
  - Write PID value
  - Set CMD.WAIT_IFS
  - Set CMD.RX_RESPONSE
  - Set CMD.TX_HEADER (trigger PID transmission) | - PID field Reception (master/slave):
  - Set CMD.RX_RESPONSE
  - Set CMD.RX_HEADER
  Waiting for transfer

- Normal Frame: N ≤ 12
- Long Frame: N ≤ 255

Note: In long frame only, not supported/implemented.
Event Trigger Method Operation (2/2)

Transmission of PID and RESPONSE by Master/Slave

Step | Software Processing | CXPI Hardware Processing
---|---|---
(2) | - | Transferring PID
(3.1) | Process PID (due to possible arbitration) | INTR.RX_HEADER_PID_DONE flag set
(3.2) | • Response Transmission:  
• Write FI value  
• Write DLEXT value\(^1\)  
• Write data to TX FIFO  
• Clear CMD.RX_RESPONSE  
• Set CMD.TX_RESPONSE  
| • Response Reception:  
• No action | • Response Reception
(4) | - | -
(5) | - | • Transmitting FI field IBS transmitter  
• Receiving FI field received  
• Transmitting DLEXT field and IBS\(^1\)  
• DLEXT field received\(^1\)
(6.1) | • Process TX FIFO\(^1\)  
• Process RX FIFO\(^1\) | • Transmitting data bytes and IBS  
• Receiving data bytes
(6.2) | - | • Data field transmission completed  
• Transmitting CRC  
• Data field reception completed  
• Receiving CRC
(7) | Frame post processing | • Response succeeded
(8) | - | End-of-Frame (EOF) 10 Tbit after response completion  
INTR.TXRX_COMPLETE flag is set

\(^1\) In long frame only
Polling Method Operation (1/2)

Transmission of PID and RESPONSE by Master/Slave, and transmission of PTYPE by Master

<table>
<thead>
<tr>
<th>Step</th>
<th>Software Processing</th>
<th>CXPI Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td><strong>Channel initialization:</strong></td>
<td>1. Channel disabled</td>
</tr>
<tr>
<td></td>
<td>• Set master/slave and modulation mode (NRZ/PWM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Automatic Transceiver Handling ON/OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Receive PID Zero Check ON (only for slave in Polling method)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• RX filtering ON/OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Set IFS length and IBS length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TX abort for bit error detection ON/OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Define PWM pulses (only PWM mode)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Configure RX sample point</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Define TX wake-up pulse length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select Time-out and configure FIFO</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Start:</strong></td>
<td>2. Entering normal mode</td>
</tr>
<tr>
<td></td>
<td>• Enable Channel and transit CXPI bus to normal mode</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td><strong>Master only:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Set CMD.WAIT_IFS</td>
<td>1. <strong>Slave only:</strong></td>
</tr>
<tr>
<td></td>
<td>• Write PID/PTYPE value</td>
<td>• Write PID value (PTYPE case only)</td>
</tr>
<tr>
<td></td>
<td>• Set CMD.RX_RESPONSE</td>
<td>• Set CMD.RX_RESPONSE</td>
</tr>
<tr>
<td></td>
<td>• Set CMD.RX_RESPONSE</td>
<td>• Set CMD.RX_HEADER</td>
</tr>
<tr>
<td></td>
<td>• Set CMD.TX_HEADER (trigger PID transmission)</td>
<td>• Set CMD.TX_HEADER</td>
</tr>
</tbody>
</table>
Polling Method Operation (2/2)

Transmission of PID and RESPONSE by Master/Slave, and transmission of PTYPE by Master

Step | Software Processing | CXPI Hardware Processing
--- | --- | ---
(2.1) | - | Transferring PTYPE (optional)
(2.2) | - | Transferring PID
(3.1) | Process PID (check for arbitration loss (slave only)) | INTR.RX_HEADER_PID_DONE flag set
(3.2) | 1. Response Transmission:  • Write FI value  • Write DLEXT value  • Write data to TX FIFO  • Clear CMD.RX_RESPONSE  • Set CMD.TX_RESPONSE | 1. Response Reception:  • No action  • Response Transmission: Waiting for TX request  • Response Reception
(4) | - | • Transmitting FI field and IBS transmitted  • Receiving FI field received
(5) | - | • Transmitting DLEXT field\(^1\) and IBS  • DLEXT field received\(^1\)
(6.1) | • Process TX FIFO\(^1\) | • Process RX FIFO\(^1\)  • Transmitting data bytes and IBS  • Receiving data bytes
(6.2) | - | • Data field transmission completed  • Transmitting CRC  • Data field reception completed  • Receiving CRC
(7) | Frame post processing | • Response succeeded
(8) | - | • Transmitting CRC  • Data field reception completed  • Receiving CRC  • Response succeeded

\(^1\) In long frame only
The CXPI channel can process NRZ signals (CTL0.MODE = 0) and PWM signals (CTL0.MODE = 1).

The following diagram shows the encoded CXPI bus signals:
Non-Return to Zero (NRZ) mode
- Since the channel does not provide the CXPI clock signal, the clock must be generated by another module separately.
Bus Signal Modulation (3/3)

› Pulse-Width Modulation (PWM) mode
› PWM encoding and decoding is done in the CXPI channel
› Additional device is not needed to generate the clock on the CXPI bus

Master/Slave in PWM mode

Master/Slave in PWM Mode

CXPI controller

IOSS

Driver/Receiver

Device boundary

3.3 V/5 V

cxpi_en_out
cxpi_tx_out
cxpi_rx_in

ten
tx
rx

CXPI bus

Master/Slave in PWM mode

Review TRM section 30.5 for additional details
FIFO Buffer

› Every channel has two separate TX and RX FIFO buffers with 16-byte depth each¹
› Provides “underflow” and “overflow” events
   – An “underflow” event is triggered by an attempt to read from an empty FIFO
   – An “overflow” event is triggered when the FIFO data is overwritten

¹ Refer to the Register TRM (INTR.TX, INTR.RX) for additional details.
P-DMA Transfer Trigger

› To avoid additional CPU access to the FIFO buffers, every CXPI channel is connected to P-DMA with trigger signal lines for both FIFO buffers
› P-DMA trigger is set in the following instances:
  - Transmission:
    - TX_FIFO_STATUS.USED < TX_FIFO_CTL.TRIGGER_LEVEL
  - Reception:
    - RX_FIFO_STATUS.USED > RX_FIFO_CTL.TRIGGER_LEVEL
Timeout Detection

The timeout feature has the following configurations:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (OFF)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

› Timeout length¹ is configured in Tbit
› TIMEOUT_SEL=1: 2 Tbits
› TIMEOUT_SEL=2: 3 Tbits

¹ You must not set IBS>TIME_LENGTH.

Hint Bar

Review TRM section 30.7.1 and Register TRM for additional details.

CXPI channel uses a fixed oversampling of 400, meaning 1 Tbit is equivalent to 400 samplings.
The timeout feature has the following configurations:

### Command Table

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAKE_TO_STANDBY</td>
<td>Direct HW to wake up from Sleep mode to Standby mode</td>
</tr>
<tr>
<td>TX_WAKE_PULSE</td>
<td>Direct HW to send wake-up pulse</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Direct HW to sleep mode</td>
</tr>
</tbody>
</table>

### Diagram

- **Sleep mode**: CXPI bus error or sleep event (internal directed or external via sleep frame). CMD.SLEEP=1 by SW and module is in idle state.
- **Standby mode**: CMD.TX_WAKE_PULSE=1 by SW
- **Normal mode**: Slave and PWM mode: Clock detected. ALL other modes: CMD.WAKE_TO_STANDBY=0 by SW

## Hint Bar

Review TRM section 30.5.3 and Register TRM for additional details. These power modes are different from the device power save modes of the device.
Enabling CXPI Channel

- Each CXPI channel must be enabled by the CTL0.ENABLED bit.
- When the same bit is cleared, all registers are cleared except the control registers.
- After it is re-enabled, the CXPI channel restarts from the Sleep mode.
Oversampling

› One CXPI bit length corresponds to 400 PCLK_CXPI_CLOCK_CH_EN cycles
› The sample moment is the sampling point at which the value of the received signal is detected for further channel processing
› The sample moment can be configured in CTL1.T_OFFSET

A dedicated internal CXPI channel clock derived from the clk_peri, which would dictate the baud rate of the CXPI functionality.
If CTL0.FILTER_EN = 1, “rx_synced” is fed to a three-input median filter and outputs the result as “rx_filtered”
CXPI Module Block Components

- **CXPI Test Registers**
  - **Test Modes**

  ![Diagram of CXPI Module Block Components]

  - From PERI/PCLK
  - clk_peri
  - PCLK_LIN_CLOCK_CH_EN[i]
  - Trigger to P-DMA
  - CXPI_TR_rx_REQ[i]
  - CXPI_TR_rx_REQ[i]
  - cxpi_interrupts_[i]_IRQn
  - TCPWM_TO_CXPI_TR[i]
  - Trigger from TCPWM IP
  - FIFO
  - CXPI Controller
  - CXPI Channel [i]
  - CXPI Channel [i]_RX
  - CXPI[i]_TX
  - CXPI[i]_EN
  - From/To IOSS/HSIOM

  *Review TRM section 30.6 for additional details*
Test Modes (1/2)

› Partial Disconnect Mode
  - Loopback mode is done via the IOSS port pin structure
  - Connection between CXPI ch.[i] and CXPI ch.[CH_NR-1]

› Advantage
  - The communication can be monitored outside the device

Copyright © Infineon Technologies AG 2020. All rights reserved.
Test Modes (2/2)

› Full Disconnect Mode
  - Full Loopback mode between CXPI ch.[i] and CXPI ch.[CH_NR-1]
  - There is no connection to port pins

› Advantage
  - Test can be performed without any external transceivers or device involvement

Hint Bar

Review TRM section 30.6.2 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>6401064</td>
<td>12/04/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6649384</td>
<td>08/07/2019</td>
<td>Added the note descriptions of all pages. Added slides 4, 16 and 18. Updated page 2, 8, 10, 11, 13, 15, 17 and 25.</td>
</tr>
<tr>
<td>*B</td>
<td>7053066</td>
<td>12/16/2020</td>
<td>Updated page 3, 22</td>
</tr>
</tbody>
</table>