Customer training workshop

TRAVEO™ T2G 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™ T2G Automotive Body Controller Entry</td>
<td>CYT2B9</td>
<td>Up to 2112KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller Entry</td>
<td>CYT2BL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster Entry</td>
<td>CYT2CL</td>
<td>Up to 4160 KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT3DL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT4DN</td>
<td>Up to 6336KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT4EN</td>
<td>Up to 6336 KB</td>
</tr>
</tbody>
</table>
Introduction to TRAVEO™ T2G Body Controller Entry

The CXPI controller is part of peripheral blocks

Review TRM chapter 30 for additional details
Introduction to TRAVEO™ T2G Cluster

The CXPI controller is part of peripheral blocks

Review TRM chapter 30 for additional details
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

Review TRM section 30.1 for additional details.
CXPI block diagram

› CXPI module block component

From PERI/PCLK
CLK_PERI

PCLK_CXPlx_CLOCK_CH_ENy

Trigger to P-DMA

CXPI_TX_TR_OUTy
CXPI_RX_TR_OUTy
cxpi_x_interrupts_y_IRQn
CXPIx_CMD_TR_INy

Trigger from TCPWM IP

Definition:
x: CXPI unit
y: CXPI Channel within a unit
CH_NR: max. channel number
i: channel = x * CH_NR + y

CLK_PERI

PCLK_CXPlx_CLOCK_CH_ENy

From/To IOSS/HSIOM

cxpi_en_out[i] / CXPIi_EN

cxpi_tx_out[i] / CXPIi_TX

cxpi_tx_in[i]
cxpi_rx_out[i]
cxpi_rx_in[i] / CXPIi_RX

AHB slave IF
Test registers

Channel registers

CXPI controller

FIFO

CLAR

Trigger to P-DMA

From PERI/PCLK

CLAR

From/To IOSS/HSIOM

cxpi_en_out[i] / CXPIi_EN

cxpi_tx_out[i] / CXPIi_TX

cxpi.tx_in[i]

cxpi.rx_out[i]

cxpi.rx_in[i] / CXPIi_RX

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 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

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

- **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

- **Frame byte:**

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

- **Long frame**

---

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

Hint Bar
Review TRM section 30.5 and Register TRM for additional details
Inter Frame Space (IFS)
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
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
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

Hint Bar

Review TRM section 30.6.7 for additional details
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'
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

<table>
<thead>
<tr>
<th>Step</th>
<th>Software Processing</th>
<th>CXPI Hardware Processing</th>
</tr>
</thead>
</table>
| (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)  | 1. PID field Transmission (master/slave):  
   • Write PID value  
   • Set CMD.WAIT_IFS  
   • Set CMD.RX_RESPONSE  
   • Set CMD.TX_HEADER (trigger PID transmission) | 2. Entering normal mode |
|      | 2. PID field Reception (master/slave):  
   • Set CMD.RX_RESPONSE  
   • Set CMD.RX_HEADER | Waiting for transfer |
## Event trigger method operation (2/2)

### Transmission of PID and RESPONSE by Master/Slave

<table>
<thead>
<tr>
<th>Step</th>
<th>Software Processing</th>
<th>CXPI Hardware Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>-</td>
<td>Transferring PID</td>
</tr>
<tr>
<td>(3.1)</td>
<td>Process PID (due to possible arbitration)</td>
<td>INTR_RX_HEADER_PID_DONE flag set</td>
</tr>
</tbody>
</table>
| (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 |
| (4)  | -                   | • Transmitting FI field IBS transmitter  
  • Receiving FI field received |
| (5)  | -                   | • 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  
  • Receiving data bytes  
  • Receiving CRC |
| (7)  | Frame post processing | • Response succeeded  
  • 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

---

**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 ON (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
  2. Entering normal mode

(1) 1. Master only:
  - Set CMD.WAIT_IFS
  - Write PID/PTYPE value
  - Set CMD.RX_RESPONSE
  - Set CMD.RX_RESPONSE
  - Set CMD.TX_HEADER (trigger PID transmission)
  1. Slave only:
    - Write PID value (PTYPE case only)
    - Set CMD.RX_RESPONSE
    - Set CMD.RX_HEADER
    - Set CMD.TX_HEADER
  Waiting for transfer
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 value1  
- 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 field1 and IBS  
- DLEXT field received1  
- Transmitting data bytes and IBS  
- Receiving data bytes
(6.1) | Process TX FIFO1  
- Process RX FIFO1  
- Data field transmission completed  
- Transmitting CRC  
- Data field reception completed  
- Receiving CRC
(6.2) | - | EOF 10 Tbit after response completion  
- INTR.TXRX_COMPLETE flag is set
(7) | Frame post processing | Response succeeded
(8) | - | Response succeeded

1 In long frame only
Bus signal modulation (1/3)

› 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

CXPI controller

IOSS

Device boundary

3.3 V/5 V

en

tx

rx

Driver/Receiver

CXPI bus

Controller

IOSS

Driver/Receiver

Device boundary

3.3 V/5 V

en

tx

rx
FIFO buffer

- Every channel has two separate TX and RX FIFO buffers with 16-byte depth each\(^1\)
- 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 length is configured in Tbit

TIMEOUT_SEL=1: 2 Tbits
TIMEOUT_SEL=2: 3 Tbits

1 You must not set IBS>TIME_LENGTH.
The timeout feature has the following configurations:

- **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

### 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>
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

Hint Bar

Review TRM section 30.6.4 and Register TRM for additional details
Oversampling

› One CXPI bit length corresponds to 400 PCLK_CXPIx_CLOCK_CH_ENy\(^1\) 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.TOFFSET

\(^1\) A dedicated internal CXPI channel clock derived from the CLK_PERI, which would dictate the baud rate of the CXPI functionality.
If CTL0i.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

From PERI/PCLK

CLK_PERI

PCLK_CXPix_CLOCK_CH_ENy

Trigger to P-DMA

CXPI_TX_TR_OUTy

CXPI_RX_TR_OUTy

cxpi_x_interrupts_y_IRQy

CXPix_CMD_TR_INy

Trigger from TCPWM IP

AHB slave IF

Test registers

Channel registers

CXPI controller

FIFO

CXPI unit

From/To IOSS/HSIOM

CXPIi_EN

CXPIi_TX

CXPIi_RX

AHB

CXPI channel

CLK_PERI

From TO IOSS/HSIOM 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]

![Diagram showing the connection between CXPI channels](image)

› Advantage
  – The communication can be monitored outside the device

Hint Bar

Review TRM section 30.7.2 for additional details
Max channel number (CH_NR)
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
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>2018/12/04</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6649384</td>
<td>2019/08/07</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>2020/12/16</td>
<td>Updated page 3, 22</td>
</tr>
<tr>
<td>*C</td>
<td>7799385</td>
<td>2022/08/17</td>
<td>Updated page 1 to 4, 6, 7, 19, 23, 27 to 29.</td>
</tr>
</tbody>
</table>