Two Wire Interface (TWI)
TWI Features

- Master mode support only
  - Master transmitter mode
  - Master receiver mode
  - All Two-wire Atmel EEPROMs Supported

- Programmable:
  - Clock baud rate Up to 400 Kbits
  - Up to Three bytes internal address (device up to 16 Mbytes)
  - Support 7-bit and 10-bit addressing

- Support fast I²C mode (up to 400kHz)
TWI Block Diagram
TWI Application

TWI Interface

Slave 1

Slave 2

TWD

TWCK

V_{DD}
Dependencies

- PMC has to be programmed 1\textsuperscript{st} for TWI to work
- PIO Controller has to be programmed for the pins to behave as intended
  - Dedicate the both as peripheral
  - Define the both line as open drain

- For example:
  - Configure TWI PIOs
  - Configure PMC by enabling TWI clock
  - Configure TWI in master mode
    - Disable interrupts
    - Reset peripheral
    - Set Master mode
  - Set TWI Clock Waveform Generator Register
    - CKDIV, CHDIV and CLDIV
TWI Clock Waveform Generator Register TWI_CWGR

- **CLDIV**: The TWCK low period $T_{\text{low}}$
- **CHDIV**: The TWCK high period $T_{\text{high}}$
- **CKDIV**: Clock divider
  - Increase the TWCK period

<table>
<thead>
<tr>
<th>18</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKDIV</td>
<td>CHDIV</td>
<td>CLDIV</td>
<td>CLDIV</td>
<td>CLDIV</td>
<td>CLDIV</td>
</tr>
</tbody>
</table>
Clock Waveform Generator

\[ T_{\text{high}} = \left( (\text{CHDIV} \times 2^{\text{CKDIV}}) + 3 \right) \times T_{\text{MCK}} \]

\[ T_{\text{low}} = \left( (\text{CLDIV} \times 2^{\text{CKDIV}}) + 3 \right) \times T_{\text{MCK}} \]

<table>
<thead>
<tr>
<th>MCK Hz</th>
<th>( T_{\text{MCK}} )</th>
<th>CKDIV 0 to 7</th>
<th>CHDIV 0 to 255</th>
<th>( T_{\text{high}} = \left( (\text{CHDIV} \times 2^{\text{CKDIV}}) + 3 \right) \times T_{\text{MCK}} )</th>
<th>CLDIV 0 to 255</th>
<th>( T_{\text{low}} = \left( (\text{CLDIV} \times 2^{\text{CKDIV}}) + 3 \right) \times T_{\text{MCK}} )</th>
<th>TWCK Second</th>
<th>Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 000 000</td>
<td>20,8E-9</td>
<td>2</td>
<td>15</td>
<td>1,3E-6</td>
<td>15</td>
<td>1,3E-6</td>
<td>2,6E-6</td>
<td>381,0E+3</td>
</tr>
</tbody>
</table>
Clock Generation

- Data are sampled on TWCK rising edge
  - Data Stable during high period of TWCK
  - Change during falling edge
  - Sampling on the rising edge

- Start and stop condition
  - TWD Falling edge while TWCK is high indicate **Start** Condition
  - TWD Rising edge while TWCK is high indicate **Stop** Condition
Control Register TWI_CR

- **START** Send a start
- **STOP** Send a stop after a complete transmission
- **MSEN = 1** = Master Mode ENABLE
- **MSDIS = 1** = Master Mode DISABLE

Both = Master mode = Disabled

- **SWRST = 1** = RESET = software controlled hardware reset
  - Writing a zero to this register has no effect
  - SWRST cleared by hardware
Master Mode Register TWI_MMR

- DADR: Device bus address (0 to 127)
  - Used to access slave devices (Hard coded)
- MREAD: Master read direction
  - Master read direction
- IADRSZ: Internal Devices address size
  - 0: No internal address (Send byte protocol)
  - 1: One-byte internal Device address size (0 to 256)
  - 2: Two-byte internal Device address size (0 to 65535)
  - 3: Three-byte internal Device address size (0 to 16 Mbytes)
**Byte protocol Data Transfer**

- Set the byte protocol excluding internal device address size
  - Master mode register \( \text{IADRSZ} = 0 \)
- Write command

```
<table>
<thead>
<tr>
<th>TWD</th>
<th>Start</th>
<th>MSB</th>
<th>DADR 7 bits</th>
<th>LSB</th>
<th>W</th>
<th>MSB</th>
<th>A</th>
<th>Command code (8bits)</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>TWCK</th>
</tr>
</thead>
</table>
```

- Read Data from Slave

```
<table>
<thead>
<tr>
<th>TWD</th>
<th>Start</th>
<th>MSB</th>
<th>DADR 7 bits</th>
<th>LSB</th>
<th>R</th>
<th>MSB</th>
<th>A</th>
<th>Data byte (8bits)</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>TWCK</th>
</tr>
</thead>
</table>
```

**S:** Start  
**W:** Write  
**R:** Read  
**A:** Acknowledge ACK  
**N:** Not Acknowledge NACK  
**DADR:** Device address (Slave)  
**IADR:** Internal device address
Internal Address Register TWI_IADR

- IADR: Internal device address (0 to 16 Mbytes)
  - Used to access slave devices internal mapping memory
Write protocol Data Transfer

- Set the byte protocol including internal device address size
  - Master mode register IADRSZ # 0
  - Internal Address Register TWI_IADR

- Write command

One byte internal Address

<table>
<thead>
<tr>
<th>TWD</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>W</th>
<th>A</th>
<th>IADR(7:0)</th>
<th>A</th>
<th>DATA(7:0)</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
</table>

Two bytes internal Address

<table>
<thead>
<tr>
<th>TWD</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>W</th>
<th>A</th>
<th>IADR(15:8)</th>
<th>A</th>
<th>IADR(7:0)</th>
<th>A</th>
<th>DATA(7:0)</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
</table>

Three bytes internal Address

|-----|---|------------|---|---|-------------|---|-----------|---|-----------|---|-----------|---|---|

S: Start
W: Write
R: Read
A: Acknowledge ACK
N: Not Acknowledge NACK
DADR: Device address (Slave)
IADR: Internal device address
Read protocol Data Transfer

- Set the byte protocol including internal device address size
  - Master mode register IADRSZ # 0
  - Internal Address Register TWI_IADR

- Read command

**One byte internal Address**

<table>
<thead>
<tr>
<th>TWD</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>W</th>
<th>A</th>
<th>IADR (7:0)</th>
<th>A</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>R</th>
<th>A</th>
<th>DATA (7:0)</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
</table>

**Two bytes internal Address**

<table>
<thead>
<tr>
<th>TWD</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>W</th>
<th>A</th>
<th>IADR (15:8)</th>
<th>A</th>
<th>IADR (7:0)</th>
<th>A</th>
<th>S</th>
<th>DADR (6:0)</th>
<th>R</th>
<th>A</th>
<th>DATA (7:0)</th>
<th>N</th>
<th>P</th>
</tr>
</thead>
</table>

**Three bytes internal Address**

|-----|---|------------|---|---|-------------|---|-------------|---|-------------|---|---|-------------|---|---|------------|---|---|

S: Start  
W: Write  
R: Read  
A: Acknowledge ACK  
N: Not Acknowledge NACK  
DADR: Device address (Slave)  
IADR: Internal device address
### TWIInterrupts

- **TWI Interrupt Enable Register TWI_IER (Write Only)**
  - 0 = No effect
  - 1 = Enable

- **TWI Interrupt Disable Register TWI_IDR (Write Only)**
  - 0 = No effect
  - 1 = Disable

- **TWI Interrupt Mask Register TWI_IMR (Read Only)**
  - 0 = Not enabled
  - 1 = Enabled

<table>
<thead>
<tr>
<th>TWI_IER, TWI_IDR, TWI_IMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACK 8</td>
</tr>
</tbody>
</table>
TWI Interrupts

- Transmission Completed
- Receive Holding Register Ready
- Transmit Holding Register Ready
- Overrun Error
- Underrun Error
- Not Acknowledge Error

1 interrupt line goes to the AIC
Read TWI_SR to determine which interrupt occurred
TWI Status Register TWI_SR

- Transmission Completed (Read And Write)
- Receive Holding Register Ready
  - 1: DATA byte has been received (Read mode)
- Transmit Holding Register Ready
  - 1: DATA byte must be transferred only (Write mode)
- Overrun Error
  - 1: Detect an overrun (Read mode)
- Underrun Error
  - 1: Detect an overrun (Write mode)
- Acknowledge
  - 1: Detect (Read & Write mode)
TWI Read Status bit

- Transmission Completed (0: during transmission)
- Receive Holding Register Ready (1: When the RHR register is full)
- Overrun Error (1: in case 2 if RXRDY=1 when other data received)
- Acknowledge (1: in case 1 if read NACK N)
  - if NACK stop the transmission and Send un Stop

Read some data at one byte size internal Address

Case
**TWI Write Status bit**

- Transmission Completed (0: during transmission)
- Transmit Holding Register Ready (1: When the THR register is Empty)
- Underrun Error (1: in case 2 if TXRDY=1 when new data must be transmit)
- Acknowledge (1: in case 1 if read NACK N)
  - if NACK stop the transmission and send a Stop

---

**Write some data at one byte size internal Address**

<table>
<thead>
<tr>
<th>TWI_THR</th>
<th>TWD</th>
<th>TWI_SR</th>
<th>TXRDY</th>
<th>TXCOMP</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write THR</td>
<td>S</td>
<td>DADR (6:0)</td>
<td>W</td>
<td>A</td>
<td>IADR (7:0)</td>
</tr>
<tr>
<td></td>
<td>MSB</td>
<td>LSB</td>
<td>MSB</td>
<td>LSB</td>
<td>MSB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TWI Acknowledge Error

- Send a bad Device address
  - Write THR start transmission
  - Send Device Address (bad address)
  - NACK (N) detection
    • Set NACK (TWI_SR)
    • Set TXRDY (TWI_SR)
- Send Stop
  • Set TXCOMP (TWI_SR)

Write some data at one byte size internal Address

<table>
<thead>
<tr>
<th>TWI_THR</th>
<th>Write THR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWI_SR</td>
<td></td>
</tr>
<tr>
<td>TWD</td>
<td></td>
</tr>
<tr>
<td>TXRDY</td>
<td></td>
</tr>
<tr>
<td>TXCOMP</td>
<td></td>
</tr>
<tr>
<td>NACK</td>
<td></td>
</tr>
<tr>
<td>Read TWI_SR</td>
<td></td>
</tr>
</tbody>
</table>
Software access

- All inline function
- Standard function is valid for AT91 core
- Register access function
  - AT91F_TWI_EnableIt
  - AT91F_TWI_DisableIt
  - AT91F_TWI_GetInterruptMaskStatus
  - AT91F_TWI_IsInterruptMasked
- Setting function
  - AT91F_TWI_CfgPIO
  - AT91F_TWI_CfgPMC
  - AT91F_TWI_Configure
Initialization

Init

Set TWI PIOs in PIO Controller

Set TWI in PMC Controller

Reset TWI (SWRST bit)
TWI_CR = 0x70

Set TWI Waveform Clock register
TWI_CWGR

Set device address
TWI_MMR
Initialization

- All inline function no cost for call
  - /* PIO AT91C_PA3_TWD & AT91C_PA4_TWCK (peripheral A)
    • AT91F_TWI_CfgPIO();

- /* Power management
  • AT91F_TWI_CfgPMC();

- /*TWI minimum Setting
  • AT91F_TWI_Configure(AT91C_BASE_TWI);

- /* Set TWI Clock (MCK = 30MHz, TWI 8KHz) CKDIV=4, CHDIV=117, CLDIV=117)
  • *AT91C_TWI_CWGR= 0x47575;

- /*Set the device address 0x55 (7 bits), addressable space device 16 bits
  • *AT91C_TWI_MMR= 0x550200;
Set Master Mode Register: Write => MREAD = 0

Set the control register
START = 1; STOP = 1; MSEN = 1

WRITE

One DATA

YES

Write THR

Start Transmission

NO

Set the control register
START = 1; MSEN = 1

Write THR

Read Status Register

TXRDY = 0

YES

Other DATA

YES

TWI_CR = STOP

Read Status Register

TXCOMP = 0

YES

END
Write to Device

- Write value 0xAA to internal device address 0x00001
  - /* Set the Internal device address
    * AT91C_TWI_IADR = 0x0001;

  - /* Set Write in mode register
    * AT91C_TWI_MMR &= 0xFFFFEFFF;

  - /* Set control register
    * AT91C_TWI_CR = AT91C_TWI_START | AT91C_TWI_MSEN | AT91C_TWI_STOP;

  - /* Set Data register for start transmission
    * AT91C_TWI_THR = 0XAA;

  - /* Wait end transmission
    * Status = *AT91C_TWI_SR ;
    * while (!(status & AT91C_TWI_TXCOMP)) {
        Status = *AT91C_TWI_SR ;
    }
Set Master Mode Register: Read => MREAD = 1

Set the control register
START = 1; STOP=1; MSEN = 1

READ

One DATA

Set the control register
START = 1; MSEN = 1

NO

Start Transmission

YES

Read Status Register

TXCOMP = 0

YES

Read RHR

YES

Read Status Register

TXRDY = 0

YES

Other DATA

TWI_CR = STOP

END

Start Transmission
Read From Device

- Read data at internal device address 0x00001
  - /* Set the Internal device address
    • *AT91C_TWI_IADR = 0x0001;

- /* Set Read in mode register
  • *AT91C_TWI_MMR |= 0x00001000;

- /* Set control register and send start
  • *AT91C_TWI_CR = AT91C_TWI_START | AT91C_TWI_MSEN | AT91C_TWI_STOP;

- /* Wait complete by TXCOMP or TXRDY
  • Status = *AT91C_TWI_SR ;
  • while (!(status & AT91C_TWI_TXCOMP)) {
    • Status = *AT91C_TWI_SR ;
  }

- /* Get data
  • Value = *AT91C_TWI_RHR;
TWI Summary

- High Speed. up to 400 K bits per second compatible Fast I2C
- Support Byte command & Internal device address protocol
- Individual Waveform clock Generator
- PIO Multiplex
- Error checking
  - Overrun, Underrun, NAK
- Master Mode only
- No Peripheral DMA support