TWIM — I²C compatible two-wire interface master with EasyDMA

TWI master with EasyDMA (TWIM) is a two-wire half-duplex master which can communicate with multiple slave devices connected to the same bus.

Listed here are the main features for TWIM:

- I²C compatible
- Supported baud rates: 100, 250, 400 kbps
- Support for clock stretching (non I²C compliant)
- EasyDMA

The two-wire interface can communicate with a bi-directional wired-AND bus with two lines (SCL, SDA). The protocol makes it possible to interconnect up to 127 individually addressable devices. TWIM is not compatible with CBUS.

The GPIOs used for each two-wire interface line can be chosen from any GPIO on the device and are independently configurable. This enables great flexibility in device pinout and efficient use of board space and signal routing.

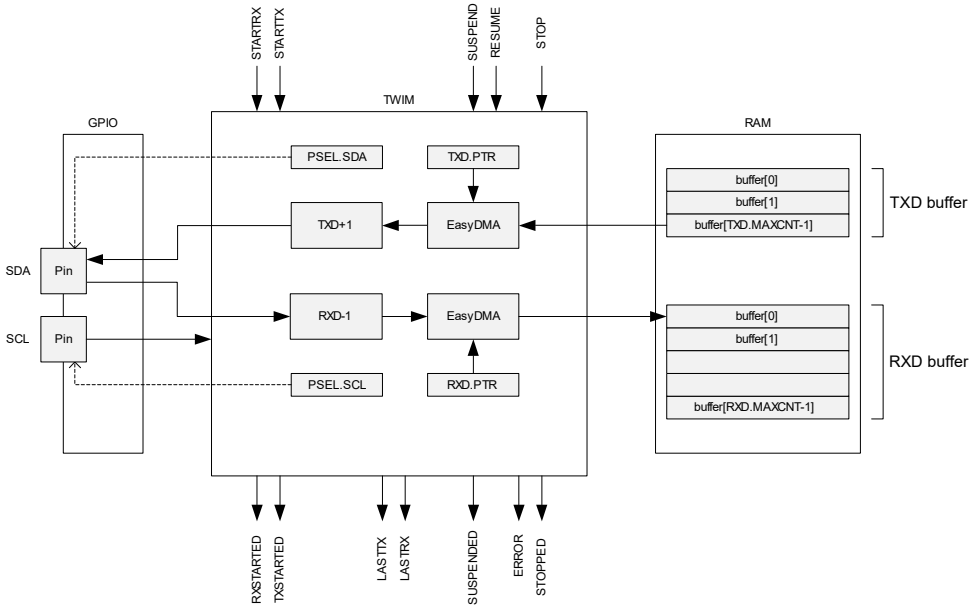

Figure 1. TWI master with EasyDMA

A typical TWI setup consists of one master and one or more slaves. For an example, see the following figure. This TWIM is only able to operate as a single master on the TWI bus. Multi-master bus configuration is not supported.

Figure 2. A typical TWI setup comprising one master and three slaves

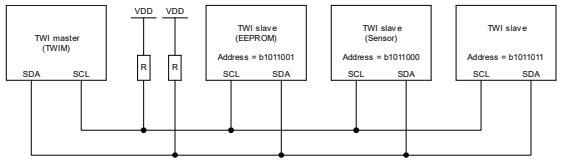

Figure 2. A typical TWI setup comprising one master and three slaves

This TWI master supports clock stretching performed by the slaves. The SCK pulse following a stretched clock cycle may be shorter than specified by the I2C specification.

The TWI master is started by triggering the STARTTX or STARTRX tasks, and stopped by triggering the STOP task. The TWI master will generate a STOPPED event when it has stopped following a STOP task.

After the TWI master is started, the STARTTX or STARTRX tasks should not be triggered again until the TWI master has issued a LASTRX, LASTTX, or STOPPED event.

The TWI master can be suspended using the SUSPEND task, this can be used when using the TWI master in a low priority interrupt context. When the TWIM enters suspend state, will automatically issue a SUSPENDED event while performing a continuous clock stretching until it is instructed to resume operation via a RESUME task. The TWI master cannot be stopped while it is suspended, thus the STOP task has to be issued after the TWI master has been resumed.

Note: Any ongoing byte transfer will be allowed to complete before the suspend is enforced. A SUSPEND task has no effect unless the TWI master is actively involved in a transfer.

If a NACK is clocked in from the slave, the TWI master will generate an ERROR event.

EasyDMA

The TWIM implements EasyDMA for accessing RAM without CPU involvement.

The TWIM peripheral implements the EasyDMA channels found in the following table.

Channel	Type	Register Cluster
TXD	READER	TXD
RXD	WRITER	RXD

Table 1. TWIM EasyDMA Channels

For detailed information regarding the use of EasyDMA, see **EasyDMA**.

The .PTR and .MAXCNT registers are double-buffered. They can be updated and prepared for the next RX/TX transmission immediately after having received the RXSTARTED/TXSTARTED event.

The STOPPED event indicates that EasyDMA has finished accessing the buffer in RAM.

Master write sequence

A TWI master write sequence is started by triggering the STARTTX task. After the STARTTX task has been triggered, the TWI master will generate a start condition on the TWI bus, followed by clocking out the address and the READ/

WRITE bit set to 0 (WRITE=0, READ=1).

The address must match the address of the slave device that the master wants to write to. The READ/WRITE bit is followed by an ACK/NACK bit (ACK=0 or NACK=1) generated by the slave.

After receiving the ACK bit, the TWI master will clock out the data bytes found in the transmit buffer located in RAM at the address specified in the TXD.PTR register. Each byte clocked out from the master will be followed by an ACK/NACK bit clocked in from the slave.

A typical TWI master write sequence is shown in the following figure. Occurrence 2 in the figure illustrates clock stretching performed by the TWI master following a SUSPEND task.

A SUSPENDED event indicates that the SUSPEND task has taken effect. This event can be used to synchronize the software.

The TWI master will generate a LASTTX event when it starts to transmit the last byte, this is shown in the following figure.

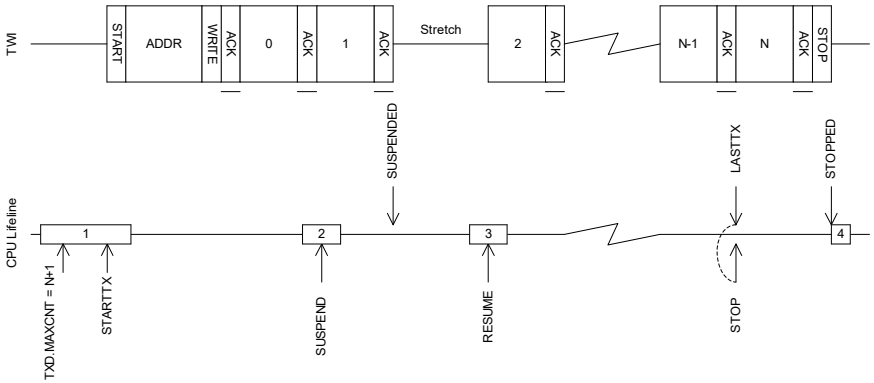

Figure 3. TWI master writing data to a slave

The TWI master is stopped by triggering the STOP task. This task should be triggered during the transmission of the last byte to secure that the TWI master will stop as fast as possible after sending the last byte. The shortcut between LASTTX and STOP can alternatively be used to accomplish this.

> **Note:** The TWI master does not stop by itself when the entire RAM buffer has been sent, or when an error occurs. The STOP task must be issued, through the use of a local or PPI shortcut, or in software as part of the error handler.

Master read sequence

A TWI master read sequence is started by triggering the STARTRX task. After the STARTRX task has been triggered, the TWI master will generate a start condition on the TWI bus, followed by clocking out the address and the READ/WRITE bit set to 1 (WRITE = 0, READ = 1). The address must match the address of the slave device that the master wants to read from. The READ/WRITE bit is followed by an ACK/NACK bit (ACK=0 or NACK = 1) generated by the slave.

After sending the ACK bit, the TWI slave will send data to the master using the clock generated by the master.

Data received will be stored in RAM at the address specified in the RXD.PTR register. The TWI master will generate an ACK after all but the last byte have been received from the slave. The TWI master will generate a NACK after the last byte received to indicate that the read sequence shall stop.

A typical TWI master read sequence is illustrated in **The TWI master reading data from a slave**. Occurrence 2 in the figure illustrates clock stretching performed by the TWI master following a SUSPEND task.

A SUSPENDED event indicates that the SUSPEND task has taken effect. This event can be used to synchronize the software.

The TWI master will generate a LASTRX event when it is ready to receive the last byte, as shown in **The TWI master reading data from a slave**. If RXD.MAXCNT > 1, the LASTRX event is generated after sending the ACK of the previously received byte. If RXD.MAXCNT = 1, the LASTRX event is generated after receiving the ACK following the address and READ bit.

The TWI master is stopped by triggering the STOP task. This task must be triggered before the NACK bit is supposed to be transmitted. The STOP task can be triggered at any time during the reception of the last byte. It is recommended to use the shortcut between LASTRX and STOP to accomplish this.

The TWI master does not stop by itself when the RAM buffer is full, or when an error occurs. The STOP task must be issued, through the use of a local or PPI shortcut, or in software as part of the error handler.

The TWI master cannot be stopped while suspended, so the STOP task must be issued after the TWI master has been resumed.

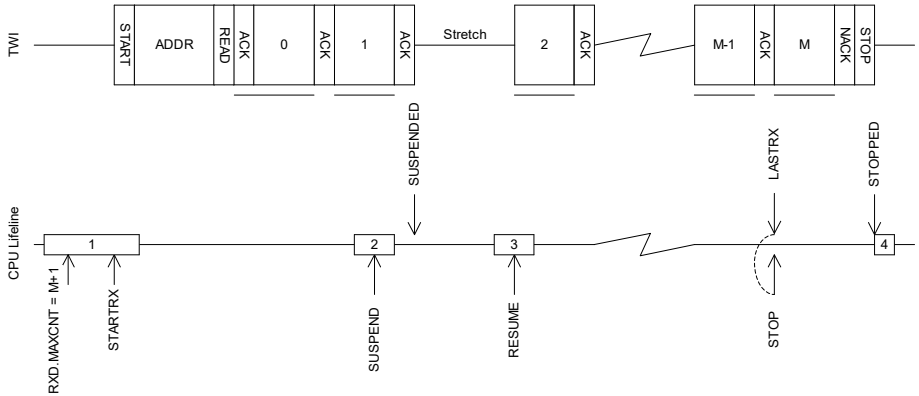

Figure 4. The TWI master reading data from a slave

Master repeated start sequence

A typical repeated start sequence is one in which the TWI master writes two bytes to the slave followed by reading four bytes from the slave. This example uses shortcuts to perform the simplest type of repeated start sequence, i.e. one write followed by one read. The same approach can be used to perform a repeated start sequence where the sequence is read followed by write.

The following figure shows an example of a repeated start sequence where the TWI master writes two bytes followed by reading four bytes from the slave.

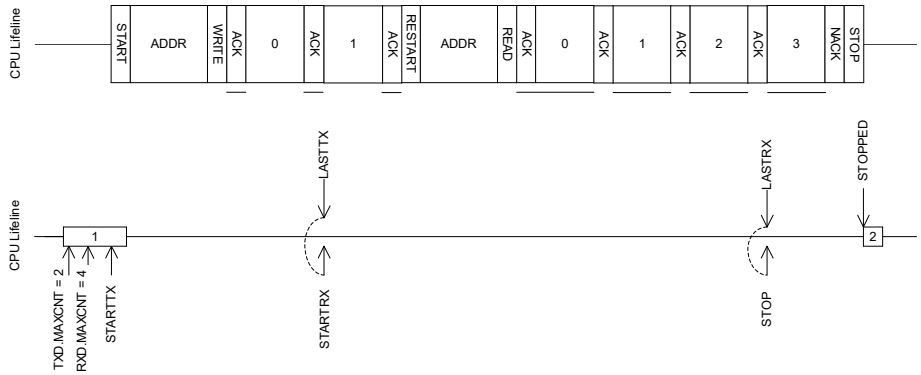

Figure 5. Master repeated start sequence

If a more complex repeated start sequence is needed, and the TWI firmware drive is serviced in a low priority interrupt, it may be necessary to use the SUSPEND task and SUSPENDED event to guarantee that the correct tasks are generated at the correct time. A double repeated start sequence using the SUSPEND task to secure safe operation in low priority interrupts is shown in the following figure.

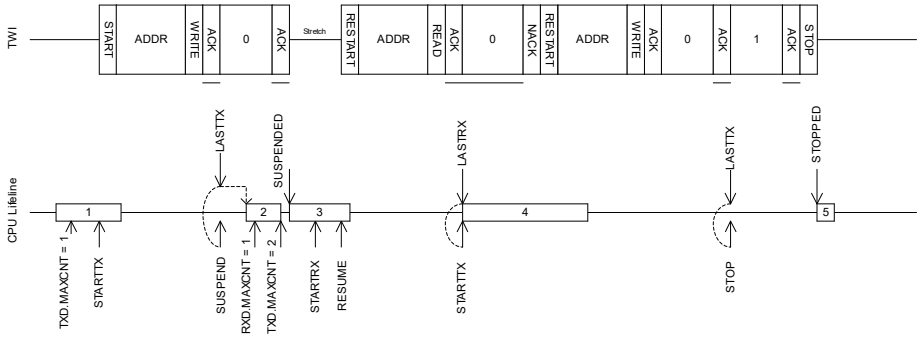

Figure 6. Double repeated start sequence

Low power

When putting the system in low power and the peripheral is not needed, lowest possible power consumption is achieved by stopping, and then disabling the peripheral.

When the STOP task is sent, the software shall wait until the STOPPED event is received as a response before disabling the peripheral through the ENABLE register. If the peripheral is already stopped, the STOP task is not required.

Master mode pin configuration

The SCL and SDA signals associated with the TWI master are mapped to physical pins according to the configuration specified in the PSEL.SCL and PSEL.SDA registers respectively.

The PSEL.SCL and PSEL.SDA registers and their configurations are only used as long as the TWI master is enabled, and retained only as long as the device is in ON mode. When the peripheral is disabled, the pins will behave as regular GPIOs, and use the configuration in their respective OUT bit field and PIN_CNF[n] register. PSEL.SCL, PSEL.SDA must only be configured when the TWI master is disabled.

To secure correct signal levels on the pins used by the TWI master when the system is in OFF mode, and when the TWI master is disabled, these pins must be configured in the GPIO peripheral as described in the following table.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behavior.

TWI master signal	TWI master pin	Direction	Output value	Drive strength
SCL	As specified in PSEL.SCL	Input	Not applicable	S0D1
SDA	As specified in PSEL.SDA	Input	Not applicable	S0D1

Table 2. GPIO configuration before enabling peripheral

Registers

Base address	Peripheral	Instance	Description	Configuration	
0x40003000	TWIM	TWIM0	Two-wire interface master 0		
0x40004000	TWIM	TWIM1	Two-wire interface master 1		

Table 3. Instances

Register	Offset	Description	
TASKS_STARTRX	0x000	Start TWI receive sequence	
TASKS_STARTTX	0x008	Start TWI transmit sequence	
TASKS_STOP	0x014	Stop TWI transaction. Must be issued while the TWI master is not suspended.	
TASKS_SUSPEND	0x01C	Suspend TWI transaction	
TASKS_RESUME	0x020	Resume TWI transaction	
EVENTS_STOPPED	0x104	TWI stopped	
EVENTS_ERROR	0x124	TWI error	
EVENTS_SUSPENDED	0x148	SUSPEND task has been issued, TWI traffic is now suspended.	
EVENTS_RXSTARTED	0x14C	Receive sequence started	
EVENTS_TXSTARTED	0x150	Transmit sequence started	
EVENTS_LASTRX	0x15C	Byte boundary, starting to receive the last byte	
EVENTS_LASTTX	0x160	Byte boundary, starting to transmit the last byte	
SHORTS	0x200	Shortcuts between local events and tasks	
INTEN	0x300	Enable or disable interrupt	
INTENSET	0x304	Enable interrupt	
INTENCLR	0x308	Disable interrupt	
ERRORSRC	0x4C4	Error source	
ENABLE	0x500	Enable TWIM	
PSEL.SCL	0x508	Pin select for SCL signal	
PSEL.SDA	0x50C	Pin select for SDA signal	
FREQUENCY	0x524	TWI frequency. Accuracy depends on the HFCLK source selected.	
RXD.PTR	0x534	Data pointer	
RXD.MAXCNT	0x538	Maximum number of bytes in receive buffer	
RXD.AMOUNT	0x53C	Number of bytes transferred in the last transaction	
RXD.LIST	0x540	EasyDMA list type	
TXD.PTR	0x544	Data pointer	
TXD.MAXCNT	0x548	Maximum number of bytes in transmit buffer	
TXD.AMOUNT	0x54C	Number of bytes transferred in the last transaction	
TXD.LIST	0x550	EasyDMA list type	
ADDRESS	0x588	Address used in the TWI transfer	

Table 4. Register overview

TASKS_STARTRX

Address offset: 0x000

Start TWI receive sequence

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0

ID	Access	Field	Value ID	Value	Description
A	W	TASKS_STARTRX			Start TWI receive sequence
			Trigger	1	Trigger task

TASKS_STARTTX

Address offset: 0x008

Start TWI transmit sequence

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0

ID	Access	Field	Value ID	Value	Description
A	W	TASKS_STARTTX			Start TWI transmit sequence
			Trigger	1	Trigger task

TASKS_STOP

Address offset: 0x014

Stop TWI transaction. Must be issued while the TWI master is not suspended.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0

ID	Access	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stop TWI transaction. Must be issued while the TWI master is not suspended.
			Trigger	1	Trigger task

TASKS_SUSPEND

Address offset: 0x01C

Suspend TWI transaction

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0

ID	Access	Field	Value ID	Value	Description
A	W	TASKS_SUSPEND			Suspend TWI transaction
			Trigger	1	Trigger task

TASKS_RESUME

Address offset: 0x020

Resume TWI transaction

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	W	TASKS_RESUME			Resume TWI transaction
			Trigger	1	Trigger task

EVENTS_STOPPED

Address offset: 0x104

TWI stopped

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_STOPPED			TWI stopped
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_ERROR

Address offset: 0x124

TWI error

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_ERROR			TWI error
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_SUSPENDED

Address offset: 0x148

SUSPEND task has been issued, TWI traffic is now suspended.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_SUSPENDED			SUSPEND task has been issued, TWI traffic is now suspended.
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_RXSTARTED

Address offset: 0x14C

Receive sequence started

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_RXSTARTED			Receive sequence started
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_TXSTARTED

Address offset: 0x150

Transmit sequence started

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_TXSTARTED			Transmit sequence started
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_LASTRX

Address offset: 0x15C

Byte boundary, starting to receive the last byte

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000	0000

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_LASTRX			Byte boundary, starting to receive the last byte
			NotGenerated	0	Event not generated
			Generated	1	Event generated

EVENTS_LASTTX

Address offset: 0x160

Byte boundary, starting to transmit the last byte

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	EVENTS_LASTTX			Byte boundary, starting to transmit the last byte
			NotGenerated	0	Event not generated
			Generated	1	Event generated

SHORTS

Address offset: 0x200

Shortcuts between local events and tasks

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					F	E D C B	A	
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	LASTTX_STARTRX			Shortcut between event LASTTX and task STARTRX
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
B	RW	LASTTX_SUSPEND			Shortcut between event LASTTX and task SUSPEND
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
C	RW	LASTTX_STOP			Shortcut between event LASTTX and task STOP
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
D	RW	LASTRX_STARTTX			Shortcut between event LASTRX and task STARTTX
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
E	RW	LASTRX_SUSPEND			Shortcut between event LASTRX and task SUSPEND
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
F	RW	LASTRX_STOP			Shortcut between event LASTRX and task STOP
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut

INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		J	I	H	G	F	D	A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	STOPPED			Enable or disable interrupt for event STOPPED
			Disabled	0	Disable
			Enabled	1	Enable
D	RW	ERROR			Enable or disable interrupt for event ERROR
			Disabled	0	Disable
			Enabled	1	Enable
F	RW	SUSPENDED			Enable or disable interrupt for event SUSPENDED
			Disabled	0	Disable
			Enabled	1	Enable
G	RW	RXSTARTED			Enable or disable interrupt for event RXSTARTED
			Disabled	0	Disable
			Enabled	1	Enable
H	RW	TXSTARTED			Enable or disable interrupt for event TXSTARTED
			Disabled	0	Disable
			Enabled	1	Enable
I	RW	LASTRX			Enable or disable interrupt for event LASTRX
			Disabled	0	Disable
			Enabled	1	Enable
J	RW	LASTTX			Enable or disable interrupt for event LASTTX
			Disabled	0	Disable
			Enabled	1	Enable

INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		J	I	H	G	F	D	A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	STOPPED			Write '1' to enable interrupt for event **STOPPED**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	ERROR			Write '1' to enable interrupt for event **ERROR**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW	SUSPENDED			Write '1' to enable interrupt for event **SUSPENDED**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW	RXSTARTED			Write '1' to enable interrupt for event **RXSTARTED**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW	TXSTARTED			Write '1' to enable interrupt for event **TXSTARTED**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	LASTRX			Write '1' to enable interrupt for event **LASTRX**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
J	RW	LASTTX			Write '1' to enable interrupt for event **LASTTX**
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID		J	I	H	G	F	D	A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	STOPPED			Write '1' to disable interrupt for event **STOPPED**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	ERROR			Write '1' to disable interrupt for event **ERROR**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW	SUSPENDED			Write '1' to disable interrupt for event **SUSPENDED**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW	RXSTARTED			Write '1' to disable interrupt for event **RXSTARTED**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW	TXSTARTED			Write '1' to disable interrupt for event **TXSTARTED**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	LASTRX			Write '1' to disable interrupt for event **LASTRX**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
J	RW	LASTTX			Write '1' to disable interrupt for event **LASTTX**
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

ERRORSRC

Address offset: 0x4C4

Error source

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								C B A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	OVERRUN			Overrun error A new byte was received before previous byte got transferred into RXD buffer. (Previous data is lost)
			NotReceived	0	Error did not occur
			Received	1	Error occurred
B	RW	ANACK			NACK received after sending the address (write '1' to clear)
			NotReceived	0	Error did not occur
			Received	1	Error occurred
C	RW	DNACK			NACK received after sending a data byte (write '1' to clear)
			NotReceived	0	Error did not occur
			Received	1	Error occurred

ENABLE

Address offset: 0x500

Enable TWIM

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	ENABLE			Enable or disable TWIM
			Disabled	0	Disable TWIM
			Enabled	6	Enable TWIM

PSEL.SCL

Address offset: 0x508

Pin select for SCL signal

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	C						B A	A A A A
Reset 0xFFFFFFFF	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

ID	Access	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
B	RW	PORT		[0..1]	Port number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

PSEL.SDA

Address offset: 0x50C

Pin select for SDA signal

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	C						B A	A A A A
Reset 0xFFFFFFFF	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

ID	Access	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
B	RW	PORT		[0..1]	Port number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

FREQUENCY

Address offset: 0x524

TWI frequency. Accuracy depends on the HFCLK source selected.

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A
Reset 0x04000000	0 0 0 0	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	FREQUENCY			TWI master clock frequency
			K100	0x01980000	100 kbps
			K250	0x04000000	250 kbps
			K400	0x06400000	400 kbps

RXD.PTR

Address offset: 0x534

Data pointer

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	PTR			Data pointer See the memory chapter for details about which memories are available for EasyDMA.

RXD.MAXCNT

Address offset: 0x538

Maximum number of bytes in receive buffer

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	MAXCNT		[0..0xFFFF]	Maximum number of bytes in receive buffer

RXD.AMOUNT

Address offset: 0x53C

Number of bytes transferred in the last transaction

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	R	AMOUNT		[0..0xFFFF]	Number of bytes transferred in the last transaction. In case of NACK error, includes the NACK'ed byte.

RXD.LIST

Address offset: 0x540

EasyDMA list type

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	LIST			List type
			Disabled	0	Disable EasyDMA list
			ArrayList	1	Use array list

TXD.PTR

Address offset: 0x544

Data pointer

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	PTR			Data pointer See the memory chapter for details about which memories are available for EasyDMA.

TXD.MAXCNT

Address offset: 0x548

Maximum number of bytes in transmit buffer

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	MAXCNT		[0..0xFFFF]	Maximum number of bytes in transmit buffer

TXD.AMOUNT

Address offset: 0x54C

Number of bytes transferred in the last transaction

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID					A A A A	A A A A	A A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	R	AMOUNT		[0..0xFFFF]	Number of bytes transferred in the last transaction. In case of NACK error, includes the NACK'ed byte.

TXD.LIST

Address offset: 0x550

EasyDMA list type

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID								A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	LIST			List type
			Disabled	0	Disable EasyDMA list
			ArrayList	1	Use array list

ADDRESS

Address offset: 0x588

Address used in the TWI transfer

Bit number	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ID							A A A	A A A A
Reset 0x00000000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

ID	Access	Field	Value ID	Value	Description
A	RW	ADDRESS			Address used in the TWI transfer

Electrical specification

TWIM interface electrical specifications

Symbol	Description	Min.	Typ.	Max.	Units
$f_{TWIM,SCL}$	Bit rates for TWIM[1]	100		400	kbps
$t_{TWIM,START}$	Time from STARTRX/STARTTX task to transmission started		1.5		µs

Two Wire Interface Master (TWIM) timing specifications

Symbol	Description	Min.	Typ.	Max.	Units
t_{TWIM,SU_DAT}	Data setup time before positive edge on SCL – all modes	300			ns
t_{TWIM,HD_DAT}	Data hold time after negative edge on SCL – 100, 250 and 400 kbps	500			ns
$t_{TWIM,HD\_STA,100kbps}$	TWIM master hold time for START and repeated START condition, 100 kbps	9937.5			ns
$t_{TWIM,HD\_STA,250kbps}$	TWIM master hold time for START and repeated START condition, 250 kbps	3937.5			ns
$t_{TWIM,HD\_STA,400kbps}$	TWIM master hold time for START and repeated START condition, 400 kbps	2437.5			ns
$t_{TWIM,SU\_STO,100kbps}$	TWIM master setup time from SCL high to STOP condition, 100 kbps	5000			ns
$t_{TWIM,SU\_STO,250kbps}$	TWIM master setup time from SCL high to STOP condition, 250 kbps	2000			ns
$t_{TWIM,SU\_STO,400kbps}$	TWIM master setup time from SCL high to STOP condition, 400 kbps	1250			ns
$t_{TWIM,BUF,100kbps}$	TWIM master bus free time between STOP and START conditions, 100 kbps	5800			ns
$t_{TWIM,BUF,250kbps}$	TWIM master bus free time between STOP and START conditions, 250 kbps	2700			ns
$t_{TWIM,BUF,400kbps}$	TWIM master bus free time between STOP and START conditions, 400 kbps	2100			ns

Figure 7. TWIM timing diagram, 1 byte transaction

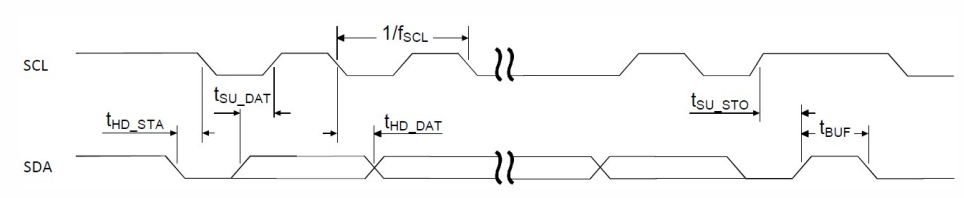

Figure 7. TWIM timing diagram, 1 byte transaction

Pullup resistor

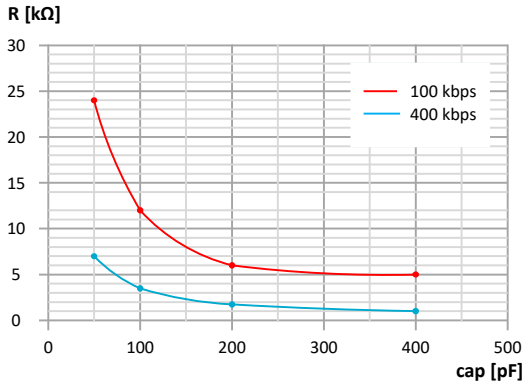

Figure 8. Recommended TWIM pullup value vs. line capacitance

- The I2C specification allows a line capacitance of 400 pF at most.
- The value of internal pullup resistor (R_{PU}) for nRF52840 can be found in GPIO — General purpose input/output.

[1] High bit rates or stronger pull-ups may require GPIOs to be set as High Drive, see GPIO — General purpose input/output for more details.