eSi-I2C

1 Contents

2 Overview

The eSi-I2C core supports the following features:

- Multi-master / slave operation.
- Clock stretching.
- 7 and 10-bit addresses.
- Programmable bit rate.
- Configurable TX and RX FIFOs.
- AMBA 3 APB slave interface.
- DMA flow-control interface.

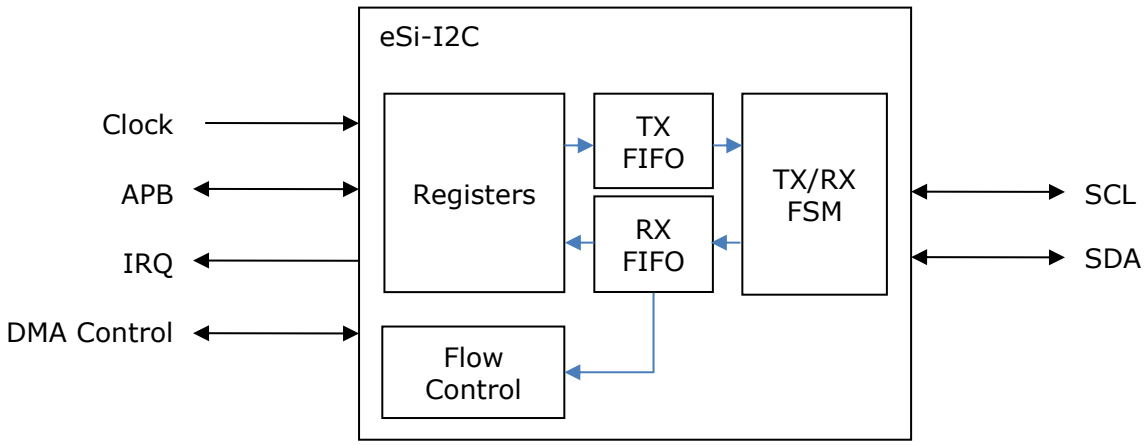

Figure 1: eSi-I2C

3 Hardware Interface

Module Name	esi_apb_i2c
HDL	Verilog
Technology	Generic
Source Files	esi_apb_i2c.v, esi_fifo.v, esi_peripheral_flow_control.v, esi_global_include.v, esi_global_cfg_include.v, esi_i2c_cfg_include.v

Configuration Option	Values	Description
SLAVE_ENABLED	TRUE, FALSE	Determines whether slave mode is supported

Table 1: Configuration Options – Defined in esi_i2c_cfg_include.v

Port	Type	Description
rx_fifo_depth	Integer	Specifies the depth of the RX FIFO. Minimum of 2 and values must be a power of 2
tx_fifo_depth	Integer	Specifies the depth of the TX FIFO. Minimum of 2 and values must be a power of 2
apb_data_width	Integer	Width of APB data bus
apb_address_width	Integer	Width of APB address bus

Table 2: Parameters

Port	Direction	Width	Description
clk	Input	1	Clock used for transmission and reception state machine. This clock must be enabled when cactive is asserted. This clock must be synchronous to pclk and clocked at the same frequency when active.
pclk	Input	1	APB clock
presetn	Input	1	APB reset, active-low
paddr	Input	apb_address_width	APB address. Only 8 LSBs are used
psel	Input	1	APB slave select
penable	Input	1	APB enable
pwrite	Input	1	APB write
pdebug	Input	1	APB noninvasive debug read
pwdata	Input	apb_data_width	APB write data
scl_in	Input	1	I2C clock in
sda_in	Input	1	I2C data in
tx_ack	Input	1	Acknowledges tx_ready after transfer complete
rx_ack	Input	1	Acknowledges rx_ready after transfer complete
cactive	Output	1	Clock active. When deasserted, clk can be gated
pready	Output	1	APB ready
prdata	Output	apb_data_width	APB read data
pslverr	Output	1	APB slave error
scl_out	Output	1	I2C clock out
scl_out_enable	Output	1	I2C clock output enable
sda_out	Output	1	I2C data out
sda_out_enable	Output	1	I2C output enable
interrupt_n	Output	1	Interrupt request, active-low

tx_ready	Output	1	Indicates device can accept new data
rx_ready	Output	1	Indicates device has data to be read

Table 3: I/O Ports

For complete details of the APB signals, please refer to the AMBA 3 APB Protocol v1.0 Specification available at:

http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

For details of the I²C-bus specification, please refer to:

http://www.nxp.com/documents/user_manual/UM10204.pdf

The I2C does not include internal synchronizing flip-flops. These should be implemented externally for the `scl_in` and `sda_in` ports if the transmitting clock domain is asynchronous to `clk`.

4 Software Interface

4.1 Register Map

Register	Address offset	Access	Description
tx_data	0x00	W	Transmit data register
rx_data	0x04	R	Receive data register
status	0x08	R/W	Status register
control	0x0c	R/W	Control register
cycles_per_bit	0x10	R/W	Cycles per bit register
address	0x14	R/W	Slave address
tx_hold_cycles	0x18	R/W	Transmit hold cycles register
rx_hold_cycles	0x1c	R/W	Receive hold cycles register
filter_cycles	0x20	R/W	Filter cycles register
txae_thresh	0x24	R/W	Transmit FIFO almost empty threshold
rxaf_thresh	0x28	R/W	Receive FIFO almost full threshold
tx_count	0x2c	R	Count of entries used in TX FIFO
rx_count	0x30	R	Count of entries used in RX FIFO

Table 4: Register Map

4.1.1 Transmit Data Register

Data to be transmitted over the I2C interface should be written to the transmit data register, in order to be written to the TX FIFO. The transmit data register should not be written to while the TXF bit in the status register is set, otherwise data loss may occur.

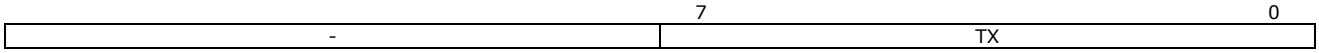

Figure 2: Format of the tx_data register

4.1.2 Receive Data Register

Data that is received over the I2C interface can be read in the receive data register. A read from the received data register will remove the data from the RX FIFO, unless pdebug is high. Setting pdebug high allows a debugger to display the first element in the RX FIFO, without affecting the program being debugged.

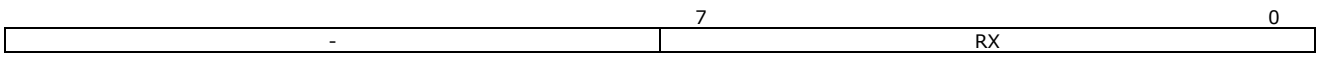

Figure 3: Format of the rx_data register

4.1.3 Status Register

The status register contains a selection of flags that indicate the current status of the I2C. To clear a bit in the status register, write a 1 to it. Writing 0 will leave it unchanged.

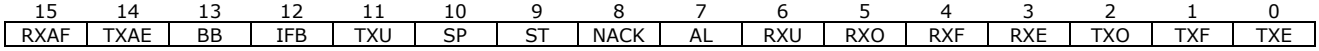

Figure 4: Format of the status register

Register	Values	Description
TXE	0 - Not empty 1 - Empty	Transmit FIFO empty
TXF	0 - Not full 1 - Full	Transmit FIFO full
TXO	0 - No overflow 1 - Overflow	Transmit FIFO overflow
RXE	0 - Not empty 1 - Empty	Receive FIFO empty
RXF	0 - Not full 1 - Full	Receive FIFO full
RXO	0 - No overflow 1 - Overflow	Receive FIFO overflow
RXU	0 – No underflow 1 – Underflow	Receive FIFO underflow
AL	0 – Arbitration not lost 1 – Arbitration lost	Arbitration lost (when master)
NACK	0 – ACK 1 – NACK	Transaction not acknowledged
ST	0 – No start 1 – Start	Start condition observed (when slave)
SP	0 – No stop 1 – Stop	Stop condition observed (when slave)
TXU	0 – No underflow 1 – Underflow	Transmit FIFO underflow
IFB	0 – Idle 1 – Busy	Indicates whether the I2C interface is busy processing a transaction or idle
BB	0 – Idle 1 – Busy	Indicates whether the I2C bus is busy in use by another master
TXAE	0 – Not almost empty 1 – Almost empty	Transmit FIFO almost empty
RXAF	0 – Not almost full 1 – Almost full	Receive FIFO almost full

Table 5: Fields of the `status` register

4.1.4 Control Register

The control register contains a selection of flags that control the operation of the I2C.

	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	GC	DC	RFSM	CS	SPIE	STIE	NIE	ALIE	RXIE	TXIE	NACK	MS	RF	E

Figure 5: Format of the `control` register

Register	Values	Description
E	0 - Disabled 1 - Enabled	Enables the I2C. When disabled, data will not be received or transmitted. The I2C should only be disabled when `status.IFB` is 0
RF	0 – Do not reset 1 – Reset FIFOs	Reset FIFOs. When written with a 1, transmit or receive FIFOs will be cleared. This bit clears automatically
MS	0 – Master 1 – Slave	Controls whether the interface operates as a master or slave. Slave mode is only supported if `SLAVE_ENABLED` is `TRUE`.

NACK	0 – ACK 1 – NACK	When operating as a slave-receiver, this flag controls whether a received byte is ACKed or NACKed
TXIE	0 - Disabled 1 - Enabled	Transmit interrupt enable
RXIE	0 - Disabled 1 - Enabled	Receive interrupt enable
ALIE	0 - Disabled 1 – Enabled	Arbitration lost interrupt enable
NIE	0 - Disabled 1 – Enabled	NACK interrupt enable
STIE	0 – Disabled 1 – Enabled	Start interrupt enable
SPIE	0 – Disabled 1 – Enabled	Stop interrupt enable
CS	0 – Disabled 1 – Enabled	When operating as a slave, the clock will be stretched if the TX FIFO is empty during a read or the RX FIFO is full during a write
RFSM	0 – Do not reset 1 – Reset FSM	Reset FSM (Finite state machine). When written with a 1, the FSM will be reset. This bit clears automatically
DC	0 – 33%/66% 1 – 50%/50%	Approximate SCL duty cycle
GC	0 – ACK general call 1 – NACK general call	Determines whether the interface will ACK transactions to the general call address (0)

Table 6: Fields of the `control` register

4.1.5 Cycles per Bit Register

The cycles per bit register is a 16-bit integer that specifies how many cycles of the clock, `clk`, the I2C clock, SCL, is held high or low for when operating as a master. Use of a 16-bit register provides support for a wide range of clock frequencies and bit rates. When operating as a slave, the bit rate is determined by the master.

When `control.DC=0`, SCL is held high for `(cycles_per_bit+filter_cycles+2)` and low for `2*(cycles_per_bit+1)`.

When `control.DC=1`, SCL is held high for `2*(cycles_per_bit+1)+filter_cycles+1` and low for `2*(cycles_per_bit+1)`.

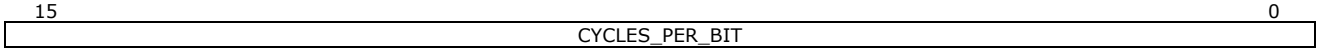

Figure 6: Format of the `cycles_per_bit` register

4.1.6 Slave Address Register

The slave address register contains the address the I2C interface will respond to (in addition to the general call address 0) when operating as a slave. If the address register is set to 0, then it will respond to any address.

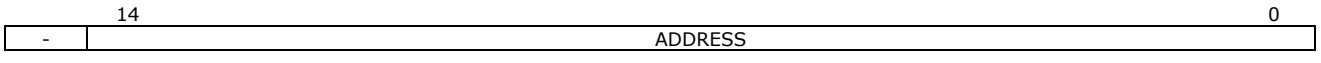

Figure 7: Format of the `address` register

7-bit addresses should be stored right aligned, with bits 14 to 7 set to 0. 10-bit addresses should also be stored right aligned, with the five most significant bits set to b11110.

4.1.7 Transmit Hold Cycles Register

The transmit hold cycles register can be used to set a minimum hold time for the SDA output enable following a falling edge on SCL. Although the hold time requirement is 0µs in the I2C specification ($t_{HD;DAT}$), other standards such as SMBus require a 300ns hold time. Additionally, some I2C devices that do not correctly implement an internal hold time for the SDA signal may benefit from a hold time being applied. The value held in the register specifies the number of `clk` cycles to hold the SDA output enable, `sda_out_enable`, following a falling edge on `scl_in`.

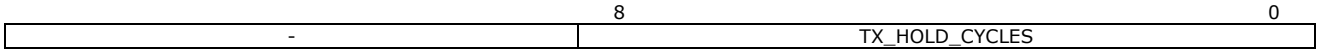

Figure 8: Format of the `tx_hold_cycles` register

4.1.8 Receive Hold Cycles Register

The receive hold cycles register can be used apply an internal hold time for the SDA input when used for detection of the START conditions. This can prevent erroneous detection of a START condition during the SCL fall time. For example, if SCL and SDA are released simultaneously, but capacitive loading on SCL results in a longer fall time for SCL than SDA, SDA may reach logic 0 while SCL is still at logic 1, which could be erroneously be interpreted as a START condition. The value held in the register specifies the number of `clk` cycles SDA is held before being evaluated by the I2C state-machine.

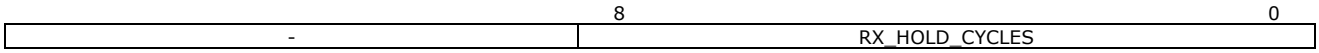

Figure 9: Format of the `rx_hold_cycles` register

4.1.9 Filter Cycles Register

The filter cycles register can be used to digitally filter glitches / spikes on the SCL and SDA inputs. The value in the register specifies the number of `clk` cycles a transition on the inputs must have been held for, before it is recognised by the internal logic.

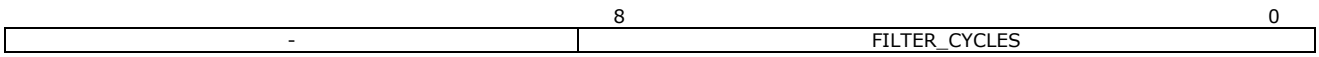

Figure 10: Format of the `filter_cycles` register

4.1.10 Transmit FIFO Almost Empty Threshold Register

The transmit FIFO almost empty threshold register sets the count of used entries in the transmit FIFO, below which, the `status.TXAE` flag and `tx_ready` will be set.

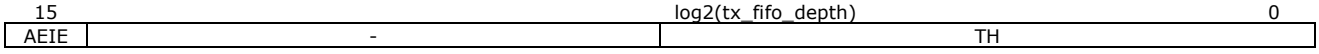

Figure 11: Format of the `txae_thresh` register

Register	Values	Description
TH	0 – tx_fifo_depth	Almost empty threshold
AEIE	0 – Interrupt disabled 1 – Interrupt enabled	Almost empty interrupt enable

Table 7: Fields of the txae_thresh register

4.1.11 Receive FIFO Almost Full Threshold Register

The receive FIFO almost full threshold register sets the count of used entries in the receive FIFO, above which, the status.RXAF flag and rx_ready will be set.

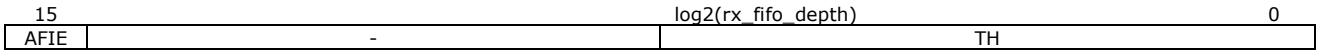

Figure 12: Format of the rxaf_thresh register

Register	Values	Description
TH	0 – rx_fifo_depth	Almost full threshold
AFIE	0 – Interrupt disabled 1 – Interrupt enabled	Almost full interrupt enable

Table 8: Fields of the rxaf_thresh register

4.1.12 Transmit FIFO Count Register

The transmit FIFO count register indicates the count of used entries in the transmit FIFO.

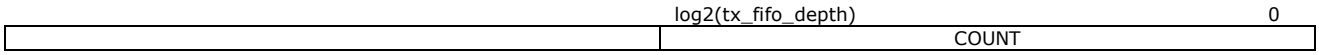

Figure 13: Format of the tx_count register

4.1.13 Receive FIFO Count Register

The receive FIFO count register indicates the count of used entries in the receive FIFO.

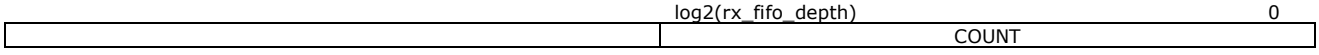

Figure 14: Format of the rx_count register

4.2 Master Transactions

4.2.1 Write Transactions

To initiate a write transaction on the I2C bus when operating as a master, two bytes of control data need to be written to the transmit data register before the data to be transmitted, as illustrated in Figure 15: Transmit Data for Write Transactions with 7-bit Addresses and Figure 16: Transmit Data for Write Transactions with 10-bit Addresses.

Byte								
1	0	0	0	SPN	NA	0	SP	ST
2	Length							
3	Address							0
4	Data 1							
...	...							
Length + 2	Data Length - 1							

Figure 15: Transmit Data for Write Transactions with 7-bit Addresses

Byte								
1	0	0	0	SPN	NA	0	SP	ST
2	Length							
3	1	1	1	1	0	Addr[9:8]		0
4	Address[7:0]							
5	Data 1							
...	...							
Length + 2	Data Length - 2							

Figure 16: Transmit Data for Write Transactions with 10-bit Addresses

The first byte is a control byte and indicates whether or not start and stop conditions are transmitted before or after the data and whether the address field is included.

- If the ST bit is set, a start condition is transmitted before the data.
- If the SP bit is set, a stop condition is transmitted after the data, providing a NACK is not received.
- If the NA bit is clear, the address field is present. If the NA bit is set, the address field is not present.
- If the SPN bit is set, a stop condition will be transmitted immediately following a NACKed byte, regardless of the ST bit.
- Other bits in this byte must be zero.

The second control byte specifies the length of the data to be transmitted, in bytes. This includes any address bytes, but does not include the two control bytes.

Following these control bytes are the address and data that will actually be transmitted on the I2C bus.

Write transactions where the data length is greater than 254 bytes (or 253 bytes when a 10-bit address is used) must be split into multiple transactions. In the first transaction, only the ST bit should be set and the SP bit should be clear, to generate a start condition, but no stop condition. In the last transaction, the SP bit should be set and the ST bit clear, which will not generate a start condition, but will generate a stop condition. If more than 509 bytes are to be written, intermediate transactions should have both the ST bit and SP clear, so that neither start nor stop conditions are generated. Only transactions that have the ST bit set should include the address field. The remaining transactions should have the NA bit set to indicate that the address field is not included.

Byte								
1	0	0	0	SPN	0	0	0	1
2	255							
3	Address							0
4	Data 1							
…	…							
257	Data 254							
258	0	0	0	SPN	1	0	0	0
259	255							
260	Data 255							
…	…							
514	Data 509							
515	0	0	0	SPN	1	0	1	0
516	1							
517	Data 510							

Figure 17: Transmit Data for Write Transactions with 510 bytes of data

If the SPN control bit is set, the master will generate a stop condition immediately following a NACKed byte, including when the SP bit is clear. If the SPN bit is clear, no stop condition will be generated following a NACKed byte, even if the SP bit is set. This allows the master to retain control of the bus and generate a repeated start condition via a new transaction with the ST bit set. It is also possible for the master to generate a stop condition manually via a new transaction with the SP bit set and the Length set to 0, as illustrated in Figure 18: Transmit Data to Generate a Stop Condition.

Byte								
1	0	0	0	0	0	0	1	0
2	0							

Figure 18: Transmit Data to Generate a Stop Condition

4.2.2 Read Transactions

To initiate a read transaction addressing a slave with a 7-bit address, two bytes of control data followed by the address need to be written to the transmit data register, as illustrated in Figure 19: Transmit Data for Read Transactions with 7-bit Addresses.

Byte								
1	0	0	0	0	NA	A	SP	ST
2	Length							
3	Address							1

Figure 19: Transmit Data for Read Transactions with 7-bit Addresses

The first byte is a control byte and indicates whether or not start and stop conditions are transmitted before or after the data.

- If the ST bit is set, a start condition is transmitted before the data.
- If the SP bit is set, a stop condition is transmitted after the data has been received.

- The A bit controls whether an ACK is generated for the last byte (as specified by the value in the Length byte). Typically this bit should be set to zero, to indicate that the last byte received should be NACKed, indicating the end of the transaction to the slave.
- If the NA bit is clear, the address field is present. If the NA bit is set, the address field is not present. Other bits in this byte should be zero.

The second control byte specifies the total length of the data in the transaction, including the address byte to be transmitted and the data bytes to be received.

After the control bytes, the address byte must be written to the transmit data register. This will then be transmitted on the I2C bus. Data received from the slave can then be read from the receive data register.

To initiate a read transaction addressing a slave with a 10-bit address, two transactions must effectively take place. First, a write transaction transmits the full 10-bit address. This write transaction is not terminated by a stop condition. Instead, a repeated start condition is generated for the second transaction, which is the actual read transaction. Only the two most-significant bits of the 10-bit slave address are transmitted. The format of the data that must be written to the transmit data register is illustrated in Figure 20: Transmit Data for Read Transactions with 10-bit Addresses.

Byte								
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	1	1	1	1	0	Addr[9:8]		0
4	Address[7:0]							
5	0	0	0	0	0	A	SP	1
6	Length							
7	1	1	1	1	0	Addr[9:8]		1

Figure 20: Transmit Data for Read Transactions with 10-bit Addresses

Read transactions where the data length is greater than 254 bytes must be split into multiple transactions. In the first transaction, the ST bit should be set and the SP bit should be clear, to generate a start condition, but no stop condition. In the last transaction, the SP bit should be set and the ST bit clear, which will not generate a start condition, but will generate a stop condition. If more than 509 bytes are to be read, intermediate transactions should have both the ST bit and SP bit clear, so that neither start nor stop conditions are generated. Only the last transaction should have the A bit set, to indicate that all but the very last byte should be ACKed. Only transactions that have the ST bit set should include the address field. The remaining transactions should have the NA bit set to indicate that the address field is not included.

Byte								
1	0	0	0	0	0	1	0	1
2	255							
3	Address							1
258	0	0	0	0	1	1	0	0
259	255							
514	0	0	0	0	1	0	1	0
515	1							

Figure 21: Transmit Data for Read Transactions with 510 bytes of data

4.2.3 NACKs and Lost Arbitration

If a NACK is received from a slave, the current transaction will be terminated and the `NACK` flag in the `status` register will be set to 1.

If arbitration is lost during a transaction, the transaction will be terminated and the `AL` flag in the `status` register will be set to 1.

In both cases, before starting a new transaction, the transmit and receive FIFOs should be cleared by setting the `RF` flag in the `control` register. Following this, the corresponding flags in the `status` register should be cleared by writing the value 1 to them.

4.2.4 Bus Clear

If a master is reset midway through a read transaction, there is the potential for bus lockup to occur if the slave was driving SDA low. This is because the slave is waiting for SCL to toggle in order to release SDA, but the master will not do this while SDA is held low. To overcome this, the bus clear transaction, as illustrated in Figure 22: Transmit Data for Bus Clear, can be used to generate 9 clocks on the bus regardless of the state of SDA. The SP bit determines whether this will be followed by a stop condition. Before starting the bus clear transaction, the `status.BB` flag may need to be cleared.

Byte								
1	0	0	1	0	0	0	SP	0
2	0	0	0	0	0	0	0	1
3	1	1	1	1	1	1	1	1

Figure 22: Transmit Data for Bus Clear

4.3 Slave Transactions

4.3.1 Write Transactions

When operating as a slave device, if a write transaction on the I2C bus is addressed to this device, as determined by the `address` register, the read/write bit and address bytes are first written to the receive FIFO, followed by the transmitted data as illustrated in Figure 23: Received Data for Write Transactions with 7-bit Addresses and Figure 24: Received Data for Write Transactions with 10-bit Addresses.

Byte		
1	Address	0
2	Data 1	
…	…	
N + 1	Data N	

Figure 23: Received Data for Write Transactions with 7-bit Addresses

Byte							
1	1	1	1	1	0	Addr[9:8]	0
2	Address[7:0]						
3	Data 1						
…	…						
N + 2	Data N						

Figure 24: Received Data for Write Transactions with 10-bit Addresses

4.3.2 Read Transactions

When operating as a slave device, if a read transaction on the I2C bus is addressed to this device, as determined by the `address` register, the read/write bit and address bytes are first written to the receive FIFO, as illustrated in Figure 25: Received Data for Read Transactions with 7-bit Addresses and Figure 26: Received Data for Read Transactions with 10-bit Addresses.

Byte		
1	Address	1

Figure 25: Received Data for Read Transactions with 7-bit Addresses

Byte							
1	1	1	1	1	0	Addr[9:8]	0
2	Address[7:0]						
3	1	1	1	1	0	Addr[9:8]	1

Figure 26: Received Data for Read Transactions with 10-bit Addresses

Data to be returned to the master should simply be written to the transmit FIFO as illustrated in Figure 27: Transmit Data for Read Transactions.

Byte	
1	Data 1
…	…
N	Data N

Figure 27: Transmit Data for Read Transactions

4.3.3 ACKs and NACKs

If when a data byte is received the receive FIFO is full and clock stretching is disabled, a NACK will be generated and the receive overflow flag will be set. For address bytes, if there is space in the receive FIFO or clock stretching is enabled, an ACK will be generated. For received data bytes from write transactions, an ACK will only be generated if the `control.NACK` flag is clear and there is space in the receive FIFO or clock stretching is enabled. For read transactions, the ACK is generated by the master, rather than the slave. If the master signals a NACK, the `status.NACK` flag will be set.

4.3.4 Clock Stretching

Clock stretching is enabled when the `control.CS` flag is set. When operating as a slave-receiver, SCL will be held low after an ACK until the received data has been written in to the receive FIFO. When operating as a slave-transmitter, SCL will be held low until data is available in the transmit FIFO, ready for transmission.

Clock stretching is not used for read transactions once `status.NACK flag` is set, as the master will signal a NACK to indicate the end of the read transaction and the slave must not stretch the clock after this, as this could prevent the master from generating the stop condition.

4.4 Interrupts

The I2C supports the following interrupts:

- Transmit interrupt
- Transmit FIFO almost empty interrupt
- Receive interrupt
- Receive FIFO almost full interrupt
- Arbitration lost interrupt
- NACK interrupt
- Start interrupt
- Stop interrupt

The transmit interrupt will be raised when the transmit FIFO is empty and the `TXIE` flag in the `control` register is set to 1. This indicates that the transmitter has no data to transmit. The transmit FIFO is read after the ACK/NACK has been received for the byte that was transmitted, so that if the transmit interrupt is raised, the NACK flag is valid for the byte that was transmitted.

The transmit FIFO almost empty interrupt will be raised when the transmit FIFO is almost empty as determined by `tx_thresh.TH` and the `AEIE` flag in the `tx_thresh` register is set to 1.

The receive interrupt will be raised when the receiver FIFO is not empty and the `RXIE` flag in the `control` register is set to 1. This indicates that the receiver has received some data.

The receive FIFO almost full interrupt will be raised when the receive FIFO is almost full as determined by `rx_thresh.TH` and the `AFIE` flag in the `rx_thresh` register is set to 1.

The arbitration lost interrupt will be raised when the `AL` flag in the `status` register and the `ALIE` flag in the `control` register are both set to 1. This indicates that arbitration was lost to another master and the transaction aborted.

The NACK interrupt will be raised when the `NACK` flag in the status register and the `NIE` flag in the `control` register are both set to 1. This indicates that a NACK was received from an I2C slave.

The start interrupt will be raised when the `ST` flag in the `status` register and the `STIE` flag in the `control` register are both set to 1. This indicates that a repeated start condition was observed on the I2C bus while operating as a slave.

The stop interrupt will be raised when the `SP` flag in the `status` register and the `SPIE` flag in the `control` register are both set to 1. This indicates that a stop condition was observed on the bus I2C while operating as a slave.

5 Revision History

Hardware Revision	Software Release	Description
1	1.0.0	Initial release
2	2.4.0	Added I2C slave support. Added `status.ST` and `status.SP` register fields. Added `control.MS`, `control.NACK` and `control.CS` register fields. Added `control.STIE` and `control.SPIE` register fields. Added `address` register. Added `tx_fifo_depth` and `rx_fifo_depth` parameters. Added start and stop interrupts. Added arbitration lost and NACK interrupts. Added `tx_ack` and `rx_ack` ports. Added support for 10-bit addressing. Added `A` flag. Increased length from 7-bits to 8-bits.
3	2.8.4	Added `status.IFB` register field. Added `status.BB` register field. Added `tx_hold_cycles` and `rx_hold_cycles` registers.
4	2.8.6	Added support for `NA` and `SPN` flags in control byte. Added `control.RFSM` register field. Added support for 0 length transactions.
5	2.9.1	Updated structure of slave `address` register to support distinguishing 7-bit and 10-bit addresses.
6	2.9.5	Added `filter_cycles` register.
7	3.2.12	Added `status.TXAE` field. Added `status.RXAF` field. Added `txae_thresh` register. Added `rxaf_thresh` register.
8	3.3.7	`rx_hold_cycles` only used for START detection.
9	3.3.9	Added `tx_count` register. Added `rx_count` register.
10	5.0.6	Added `control.DC` field.
11	6.0.2	Added `pdebug` input.
12	6.0.3	`tx_ready` is driven by TX FIFO almost empty, rather than not full. `rx_ready` is driven by RX FIFO almost full, rather than not empty.
13	6.0.3	Added `control.GC` field. Added bus clear transaction.

Table 9: Revision History

