XC800 Family

AP08099

Using the IIC Module in the XC800 Family

Application Note

V1.0, 2010-02

Microcontrollers

Edition 2010-02

Published by
Infineon Technologies AG
81726 Munich, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XC800 Family
Revision History: V1.0 2010-02

Previous Version(s):

Page	Subjects (major changes since last revision)

Trademarks

TriCore® is a trademark of Infineon Technologies AG.

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

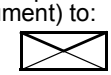

Table of Contents

1 Introduction

The Inter-Integrated Circuit (IIC) module in the XC800 family can operate in either master or slave mode. In master mode the IIC module performs arbitration, to allow it to operate in multi-master systems. The module provides communication at data rates of up to 400 KBaud, and features both 7-bit and 10-bit addressing.

This application note describes the set-up and use of the IIC module in both master mode and slave mode; outlining its use as a master transmitter/receiver and slave transmitter/receiver. Exception handling during master and slave modes is also discussed.

Finally, an application example that performs a data loopback from the master to slave and back to master, is provided to demonstrate the usage of the IIC module in each of these modes.

2 IIC Bus Overview

The Inter-Integrated Circuit (IIC) bus allows integrated circuits to communicate directly with each other via a simple bi-directional 2-wire bus. The two bus lines are serial clock line (SCL), and serial data line (SDA). The IIC bus is a standard bus system used in consumer electronics, telecommunications, and industrial electronics. The following paragraphs provide a brief overview of the IIC bus. For details and electrical characteristics, refer to the IIC protocol specifications.

A HIGH-to-LOW transition of the data line (SDA) while the clock line (SCL) is HIGH indicates a START condition. A LOW-to-HIGH transition of the SDA while SCL is HIGH defines a STOP condition. The data line can only be changed when the clock signal on the SCL line is LOW. Therefore, the data on the SDA line must be stable during the HIGH period of the clock signal. The bus is considered to be busy after the Start condition and is considered to be free at a certain time interval after the STOP condition.

Information on the SDA line must be 8 bits long. The data is transferred serially with the most significant bit first, followed by an acknowledge bit. The 9th clock pulse of the acknowledge bit is generated by the master. The transmitting device has to release the SDA line (HIGH or in the high impedance state) during this clock pulse while the device that needs to acknowledge has to pull down the SDA line during this clock pulse. The number of data bytes transferred between the START and STOP condition from the transmitter and receiver is not limited.

The receiver is obliged to generate an acknowledge bit after each byte of data that has been received. When the receiver does not provide an acknowledge bit after having received a byte of data, the data line must be left HIGH or in the high impedance state by the slave. The master can then generate a STOP condition to abort the transfer. One of the reasons for the receiver not providing the acknowledge bit is that the receiver is performing some real-time function. If the master is receiving data, it must signal the end of the data transfer to the slave, by not generating an acknowledge bit on the last byte of data received. The slave must then release the data line to allow the master to generate the STOP condition.

A complete data transfer format in 7-bit addressing mode is shown in **Figure 1**. After a START condition, a slave address or General Call Address (0x00) is sent. The address is 7 bits long followed by an 8th bit which is a data direction bit (R/W). A 0 for data direction bit indicates a transmission (WRITE), and a 1 indicates a request for data (READ).

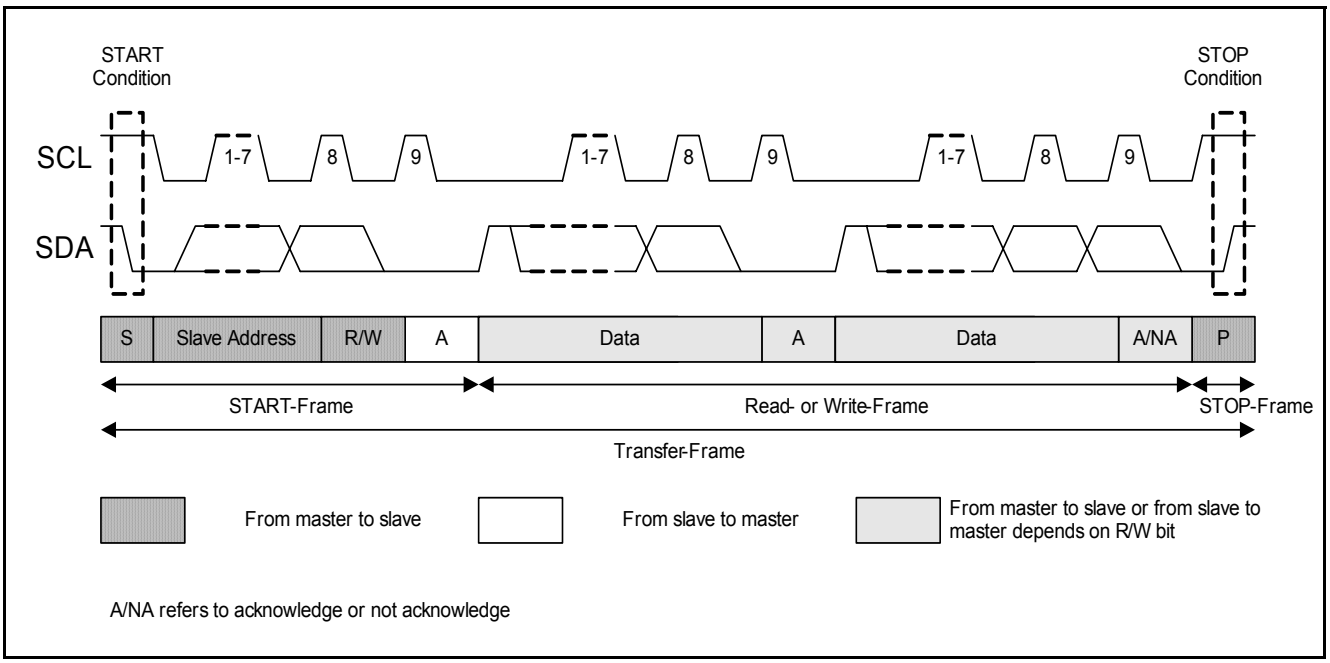

Figure 1 7-Bit Addressing Mode Timing Diagram

Figure 2 shows the IIC bus data transfer format of writing data from the master to slave device and of reading data from the slave device.

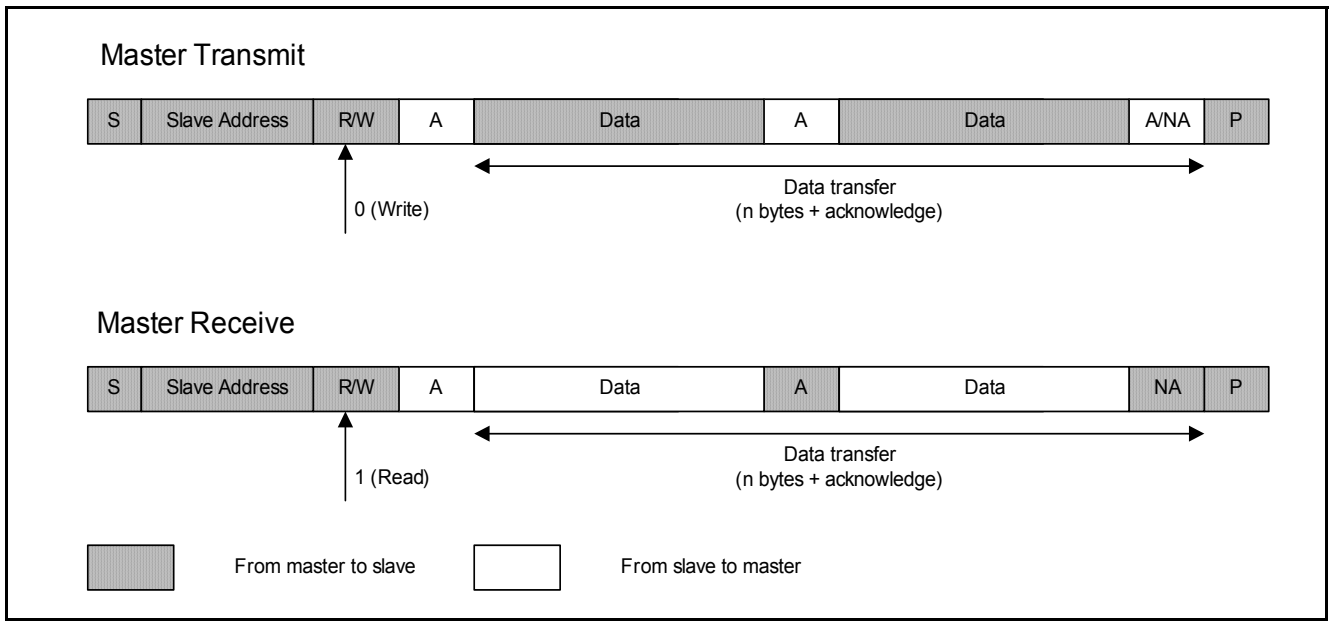

Figure 2 7-Bit Addressing Mode Data Transfer Format

A data transfer is always terminated by a STOP condition generated by the master. However, if the master still wishes to communicate on the bus, it can generate a repeated START condition and address the same device or another slave device without first generating a STOP condition. This combined data transfer format is shown in **Figure 3**.

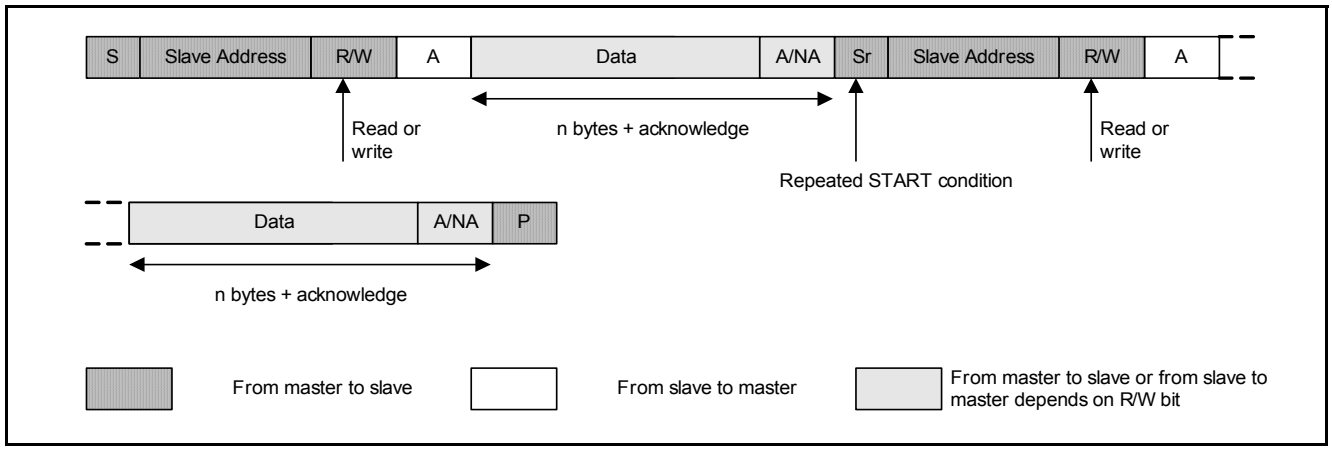

Figure 3 7-Bit Addressing Mode Data Transfer Format With Repeated START

The IIC protocol also provides a 10-bit addressing mode to support more slave addresses. A complete data transfer format in 10-bit addressing mode is shown in **Figure 4**. After a START condition, the first byte of the slave address is sent. The first seven bits of the first byte must take the pattern $11110XX_B$, of which the last two bits (XX) are the two Most-Significant Bits (MSBs) of the 10-bit address. The 8th bit is a data direction bit (R/W), which in this case must be selected as 0 (Write) since the master device will be sending the second byte of the slave address.

After the slave device has acknowledged the second address byte, a master transmitter can proceed to send the data byte(s), as with the 7-bit addressing mode. A master receiver on the other hand, needs to issue a repeated START condition and resend the first address byte with the data direction bit set to 1 (Read). The slave device can then proceed to send the data byte(s).

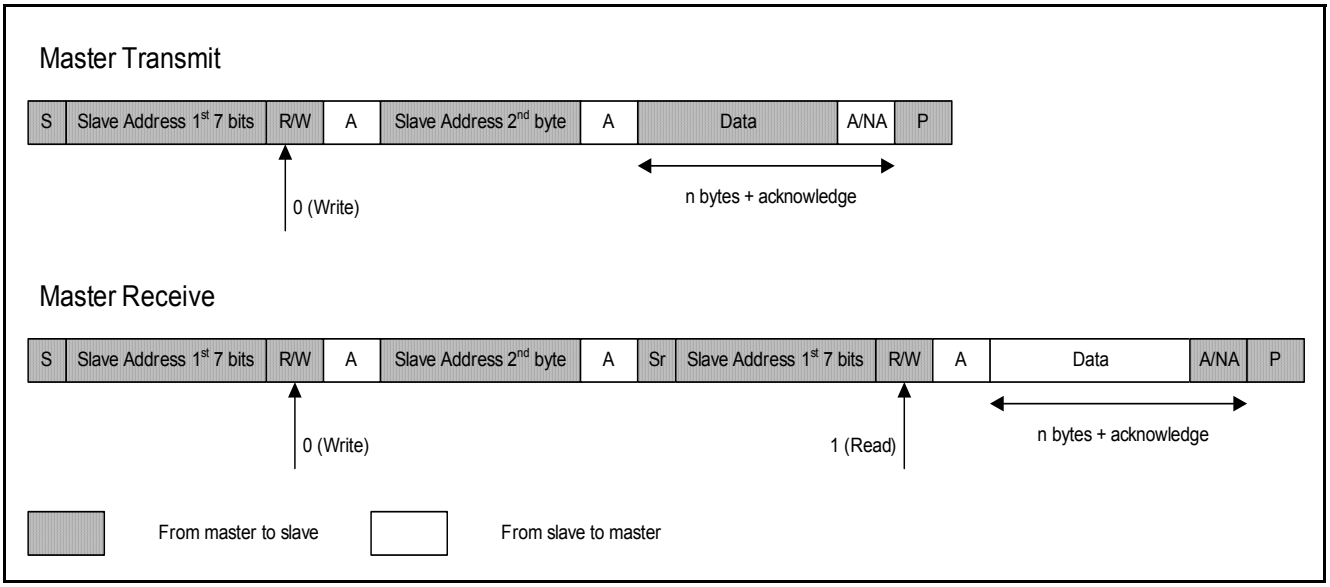

Figure 4 10-Bit Addressing Mode Data Transfer Format

3 Configure IIC to Master Mode

To configure the IIC module to master mode, the following three steps are required:

- Initialize the port registers to set up SCL and SDA pins (see **Section 3.1**)
- Initialize the Baud Rate Control Register IIC_BRCR to configure the IIC baud rate (see **Section 3.2**)
- Initialize the IIC Control Register IIC_CNTR to enable the IIC module and if interrupt is used, to enable the IIC interrupt (see **Section 3.3**)

3.1 Set Up SCL and SDA Pins

SCL and SDA are bi-directional pins and can be selected from several pin option sets in the XC800 device. The selection is made through the initialization of the Port x Alternate Output Function Select Registers Px_ALTSELy.Pn to the corresponding value.

The open drain mode and pull-ups for the selected SCL and SDA pins have to be enabled as well. This is done by setting the corresponding Pn bits in Port x Open Drain Control (Px_OD), Port x Pull-up/Pull-down Select and Enable (Px_PUDSEL and PxPUDEN) registers to 1.

Note: If more than one set of input/output are selected via Px_ALTSELn register, the set with the lower set number has the highest priority. For example, if output signal SDA_0 and SDA_1 are selected at the same time, the input will only come from SDA_0.

To provide an example, the IIC pin function definition in XC800 Family is shown in **Table 1**.

Table 1 IIC Pin Functions in XC822/824

Pin	Function	Desciption	Selected By
P0.4	SCL_0	IIC Clock Input/Output	P0_ALTSEL0.P4 = 0_B P0_ALTSEL1.P4 = 0_B P0_ALTSEL2.P4 = 1_B
P0.2	SCL_1	IIC Clock Input/Output	P0_ALTSEL0.P2 = 0_B P0_ALTSEL1.P2 = 1_B
P0.6	SDA_0	IIC Data Input/Output	P0_ALTSEL0.P6 = 0_B P0_ALTSEL1.P6 = 0_B P0_ALTSEL2.P6 = 1_B
P0.3	SDA_1	IIC Data Input/Output	P0_ALTSEL0.P2 = 0_B P0_ALTSEL1.P2 = 1_B

If SCL_1 and SDA_1 are selected as the active pins, the following code will be required to initialize the various port registers.

```
/// -----------------------------
/// SCL and SDA pin configuration
/// -----------------------------
/// - P0.2 (SCL_1) is selected as SCL pin
/// - P0.3 (SDA_1) is selected as SDA pin
/// - SCL and SDA pins are configured as open drain and pull up device

SFR_PAGE(_pp2, noSST); // switch to page 2
P0_ALTSEL0 &= ~(ubyte) 0x0C; // clear the P0_ALTSEL0 bits
P0_ALTSEL1 |= 0x0C; // set the P0_ALTSEL1 bits

SFR_PAGE(_pp3, noSST); // switch to page 3
```

```
P0_OD |= 0x0C; // set the P0_OD bits

SFR_PAGE(_pp1, noSST); // switch to page 1
P0_PUDEN |= 0x0C; // set the P0_PUDEN bits
P0_PUDSEL |= 0x0C; // set the P0_PUDSEL bits
```

3.2 Configure IIC Baud Rate

Two dividers are implemented in the IIC module to generate the required baud rate on the IIC bus from the module clock (PCLK). They are defined by the PREDIV and BRP bits in the write-only Baud Rate Control Register (IIC_BRCR). The baud rate can be calculated based on the following equation:

(1)

$$\text{Baud rate} = \frac{\text{PCLK}}{2^{\text{PREDIV}} \times (\text{BRP} + 1) \times 10}$$

Note: To ensure the correction detection of the START and STOP conditions on the bus, the IIC module must sample the bus at least 10 times faster than the bus clock speed of the fastest master on the bus. The sampling rate is given by the equation PCLK / 2^{PREDIV}.

Some commonly used baud rates together with the required divider settings are shown in **Table 2**.

Table 2 Baud Rate Selection

PCLK	Baud Rate = 100 KBaud		Baud Rate = 400 KBaud	
	PREDIV	BRP+1	PREDIV	BRP+1
8 MHz	1 (1_H)	4 (4_H)	1 (1_H)	1 (1_H)
24 MHz	1 (1_H)	12 (C_H)	1 (1_H)	3 (3_H)

The following code shows the initialization of the BRCR register for PCLK = 24 MHz and baud rate = 400 Kbaud.

```
/// -----------------------------
/// IIC module baud rate setting
/// -----------------------------
/// - Input clock of the IIC module (PCLK) is 24 MHz
/// - Baud rate = 400 Kbaud

IIC_BRCR = 0x11; // BRP = 2 and PREDIV = 1
```

3.3 Enable IIC Module and Interrupt

The IIC module is enabled by setting the bit ENAB in the IIC Control Register (IIC_CNTR) to 1. If interrupt is used to monitor the state of IIC instead of the polling method, the bit IIC_CNTR.IEN should also be set to 1.

The following code shows the initialization of the IIC_CNTR register.

```
/// --------------------------
/// IIC module control setting
/// --------------------------
/// - IIC is enabled
/// - IIC interrupt is selected
```

```
IIC_CNTR = 0xC0; //load control register
```

Note: In order to enable the IIC interrupt generation, the bit EX2 in Interrupt Node Enable Register 1 (IEN1) needs to be enabled.

4 Operate IIC in Master Mode

The IIC module uses a 5-bit status code to indicate its state at any one time. The status code can be read by software through the read-only Status Register IIC_STAT. Besides the status code $F8_H$, which indicates the idle state, all other states will set the IFLG bit to 1 and generate the IIC interrupt if interrupt is enabled.

A software driver is therefore required to process the current state of the IIC module and set it up for the next expected state. A general overview of the IIC master mode operation is shown in **Figure 5**.

Note: Other implementations of the IIC master mode operation are possible.

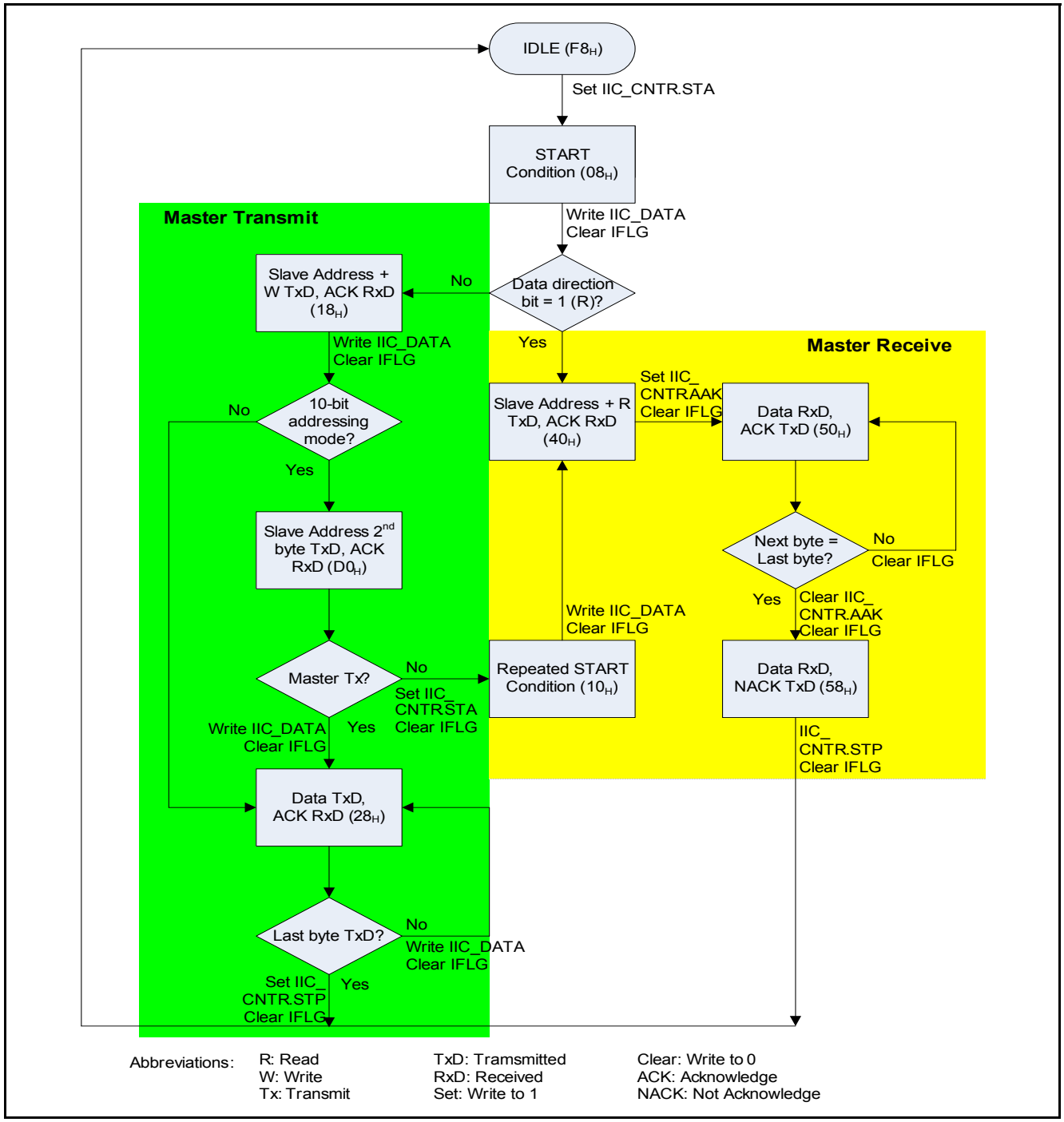

Figure 5 IIC Master Operation General Overview

The list of status codes associated with IIC master mode operation is shown in Table 3.

Table 3 Status Codes used in Master Mode Operation

IIC_STAT Register	Status
08_H	START condition transmitted
10_H	Repeated START condition transmitted
18_H	Address and write bit transmitted, ACK received
28_H	Data byte transmitted in master mode, ACK received
40_H	Address and read bit transmitted, ACK received
50_H	Data byte received in master mode, ACK transmitted
58_H	Data byte received in master mode, ACK not transmitted
$D0_H$	Second address byte and write bit transmitted, ACK received
$F8_H$	No relevant status information, IFLG = 0

4.1 Generate START, Repeated START and STOP Conditions

When the IIC_CNTR.STA bit is set to 1, the IIC module will enter master mode and transmit a START condition once the IIC bus is free. If the IIC module is already in master mode and one or more bytes have been transmitted, a repeated START condition will be transmitted. If the IIC module is still being accessed in slave mode, it will complete the data transfer in slave mode and enter master mode once the IIC bus has been released.

When the IIC_CNTR.STP bit is set to 1, the IIC module will transmit a STOP condition. If both STA and STP bits are set, the IIC module will first transmit a STOP condition (if in master mode) before transmitting a START condition.

Note: Both STA and STP bits are automatically cleared after the corresponding condition has been sent.

4.2 Generate ACK / NACK

For a IIC master receiver to generate an acknowledgement bit (ACK) following the reception of a data byte from a slave transmitter, the IIC_CNTR.AAK bit must be set to 1. In the same manner, for the master to generate a not acknowledge bit (NACK), the AAK bit must be cleared to 0.

5 Exception Handling in Master Mode

The IIC module is able to detect the occurrence of exceptions during operation in master mode. The list of exceptions and their assigned status codes is shown in **Table 4**. They can be classified into the following three exception types:

- Bus Error (see **Section 5.1**)
- NACK received (see **Section 5.2**)
- Lost arbitration (see **Section 5.3**)

Table 4 Exception Status Codes in Master Mode

IIC_STAT Register	IIC Status	Exception Type
00_H	Bus error	Bus error
20_H	Address and write bit transmitted, ACK not received	NACK received
30_H	Data byte transmitted in master mode, ACK not received	NACK received
38_H	Arbitration lost in address or data byte	Lost arbitration
48_H	Address and read bit transmitted, ACK not received	NACK received
68_H	Arbitration lost in address as master, slave address and write bit received, ACK transmitted	Lost arbitration
78_H	Arbitration lost in address as master, general call address received, ACK transmitted	Lost arbitration
$B0_H$	Arbitration lost in address as master, slave address and read bit received, ACK transmitted	Lost arbitration
$D8_H$	Second address byte and write bit transmitted, ACK not received	NACK received

5.1 Bus Error

A bus error occurs if there is an illegal condition on the IIC bus. To recover from this state, the IIC_CNTR.STP bit must be set and the IIC_CNTR.IFLG bit cleared. The IIC module will then return to the idle state. There is no actual STOP condition transmitted on the bus in this case.

To request resumption of transmission, the IIC_CNTR.STA bit can be set to 1 at the same time as the STP bit is set. This will cause the IIC module to send a START condition upon recovery from the bus error.

5.2 NACK Received

During the course of the address or data byte transmission, the master IIC may receive a NACK from the slave device. This could arise if the slave device is not ready to receive the transmitted data or it receives data that it does not understand.

The way to handle this exception has to be defined by the application software. For example, the application software can log the exception as an error and send a STOP condition, or restart the transmission by sending a repeated START condition.

5.3 Lost Arbitration

Lost arbitraion may occur in a multi-master IIC bus. In master mode, the IIC module will check that each transmitted logic 1 appears on the IIC bus as a logic 1. If another device on the bus overrules and pulls the SDA line low, arbitration is lost.

If arbitration is lost during the transmission of a data byte or a NACK bit, the IIC module will return to idle state. If arbitration is lost during the transmission of an address, the IIC module will switch to slave mode so that it can recognize its own slave address or the general call address.

To request resumption of transmission, the IIC module has to send a START condition and repeat the bus arbitration when the bus is free again.

6 Configure IIC to Slave Mode

To configure the IIC module to slave mode, the same three steps to configure to master mode are required (see **Chapter 3**):

- Initialize the port registers to set up SCL and SDA pins
- Initialize the Baud Rate Control Register IIC_BRCR to configure the IIC baud rate
- Initialize the IIC Control Register IIC_CNTR to enable the IIC module and if interrupt is used, to enable the IIC interrupt

Additionally, it is also required to set up the slave address:

- Initialize the Slave Address Register IIC_ADDR (and Extended Slave Address Register IIC_ADDRX if 10-bit addressing mode is used) to define the slave address and if required, to enable the General Call Address feature (see **Section 6.1**)

Note: A device which can act as a master or slave in a multi-master IIC bus should be initialized similar to a slave device.

6.1 Define Slave Address and General Call Address Option

The 7-bit slave address is defined in the SLA bit field of the Slave Address Register IIC_ADDR. For a 10-bit slave address, the pattern 11110_B followed by the two MSB bits of the address need to be written to SLA. The remaining 8 address bits are then defined in the Extended Slave Address Register IIC_ADDRX.

The IIC module also supports the General Call Address feature of the protocol for the slave device. To enable the General Call Address feature, the bit IIC_ADDR.GCE should be set to 1.

The following two code snippets provide the example for initializing a slave device with the address $4A_H$ in 7-bit and 10-bit addressing modes. The Generall Call Address feature is also enabled in both cases.

```
/// ----------------------------------------
/// IIC Module  7-bit Slave address setting
/// ----------------------------------------
/// - Slave address is : 0x4A
/// - General call address option is enabled

IIC_ADDR = 0x95;

/// ----------------------------------------
/// IIC Module  10-bit Slave address setting
/// ----------------------------------------
/// - Slave address is : 0x04A
/// - General call address option is enabled

IIC_ADDR = 0xF1;
IIC_ADDRX = 0x4A;
```

7 Operate IIC in Slave Mode

The IIC module provides a 5-bit status code to indicate its cuurent state. The status code can be read by software through the read-only Status Register IIC_STAT. Besides the status code $F8_H$, which indicates the idle state, all other states when entered will set the bit IFLG and generate the IIC interrupt if interrupt is enabled.

In slave mode, the IIC module will hold the SCL line low after each byte has been transferred until IFLG has been cleared in the CNTR register.

A software driver is therefore required to process the current state of the IIC module and set it up for the next expected state. A general overview of the IIC slave mode operation is shown in **Figure 6**.

Note: Other implementations of the IIC slave mode operation are possible.

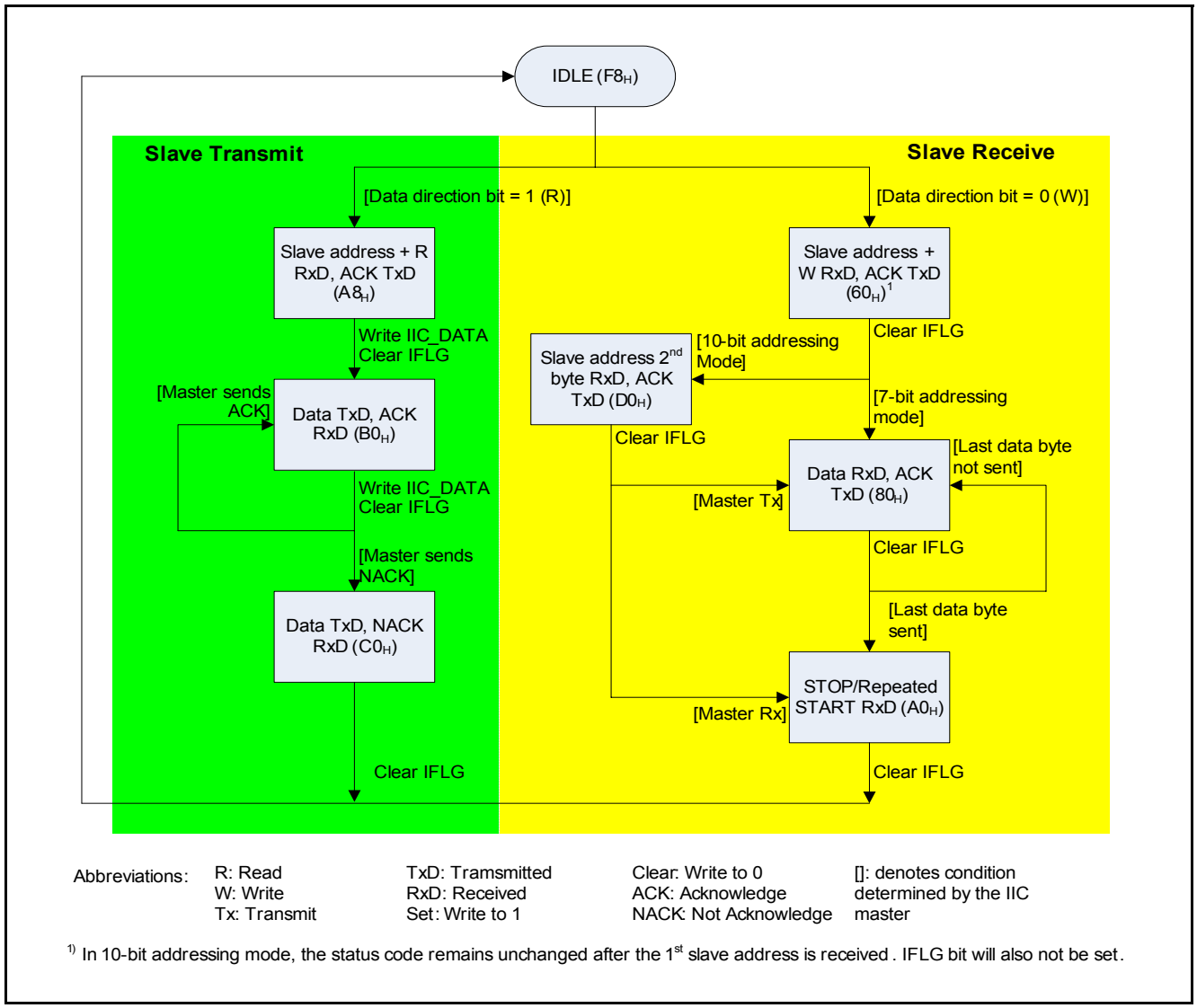

Figure 6 IIC Slave Operation General Overview

The list of status codes associated with IIC slave mode operation is shown in Table 5.

Table 5 Status Codes used in Slave Mode Operation

IIC_STAT Register	Status
60_H	Slave address and write bit received, ACK transmitted
70_H	General call address received, ACK transmitted
80_H	Data byte received after slave address received, ACK transmitted
88_H	Data byte received after slave address received, ACK not transmitted
90_H	Data byte received after general call address received, ACK transmitted
98_H	Data byte received after general call address received, ACK not transmitted
$A0_H$	STOP or repeated START mode received in slave mode
$A8_H$	Slave address and read bit received, ACK transmitted
$B8_H$	Data byte transmitted in slave mode, ACK received
$C0_H$	Data byte transmitted in slave mode, ACK not received
$C8_H$	Last byte transmitted in slave mode, ACK received
$F8_H$	No relevant status information, IFLG = 0

7.1 Generate ACK / NACK

For the IIC slave to respond to a matching 7-bit, 10-bit or General Call slave address and data byte received from the master transmitter with an acknowledge bit (ACK), the control bit AAK in IIC_CNTR register must be set to 1. If the AAK bit is cleared to 0 during a transfer, the IIC slave will transmit a NACK after the next byte is received, and set the IFLG bit. Upon clearing the IFLG bit to 0, the IIC will return to idle state.

Note: If the AAK bit is cleared to 0 in slave transmitter mode, the data byte in the IIC_DATA register is assumed to be the last byte in the transmission.

8 Exception Handling in Slave Mode

The IIC module is able to detect the occurrence of the two exceptions, bus error and NACK received (for a transmitted byte that is not the last byte), during the operation in slave mode. The corresponding status codes are shown in **Table 6**.

Table 6 Exception Status Codes in Slave Mode

IIC_STAT Register	IIC Status	Exception Type
00_H	Bus error	Bus error
$C0_H$	Data byte transmitted in slave mode, ACK not received	NACK received

8.1 Bus Error

A bus error occurs if there is an illegal condition on the IIC bus. To recover from this state, the IIC_CNTR.STP bit must be set and the IIC_CNTR.IFLG bit cleared. The IIC module will then return to the idle state. There is no actual STOP condition transmitted on the bus in this case.

8.2 NACK Received

A master receiver will send a NACK after receiving the last byte of data from the slave transmitter before sending a STOP condition. However, if the slave IIC receives a NACK for a transmitted data byte that is not the last byte, an exception is considered to have occurred.

The IIC module will interpret the NACK from the master device as a call to end transmission and after clearing the IIC_CNTR.IFLG bit, it will return to the idle state. If required, the software application can implement additional checks to determine if the complete transmission is performed, and if not, to carry out necessary recovery steps.

9 IIC Application Example

A data loopback application example, which is provided together with this application note, demonstrates the use of the IIC module in both master and slave modes.

The application example is implemented and verified using the following hardware and software components:

- 2 x XC822M Easykit boards (1 x master, 1 x slave)
- Infineon DAvE tool for code generation
- Infineon DAS client for debugging
- Keil MicroVision4 for code compilation
- Docklight V1.9 (evaluation) for use as the hyper terminal in PC host

The self-extracting EXE file accompanying the application note, contains two separate software projects; one for the master device (iic_mas_int.dpt) and one for the slave device (iic_sla_int.dpt). The master IIC application requires the PC host to send one byte of user-defined data to it, through the UART port. It then sets up a two byte data transmit buffer consisting of the received byte and the received byte incremented by one, before initiating a master transmit operation to send these two data bytes to the slave IIC.

The slave IIC application further increments both of the received bytes by one and prepares them for transmission back to the master IIC. The master IIC will then initiate a master receive operation to read these two data bytes back from the slave.

To check that the receive bytes contain the correct values, the master IIC application sums the two data values and outputs the result back to the PC host. The result obtained is the sum of two times user-defined input data and 3, see **Equation (2)**. For example, if the user-defined input data is 0x0F, the result will be 0x21 as shown in the captured screenshot in **Figure 7**.

(2)

$$Result = 2 \times Data + 3$$

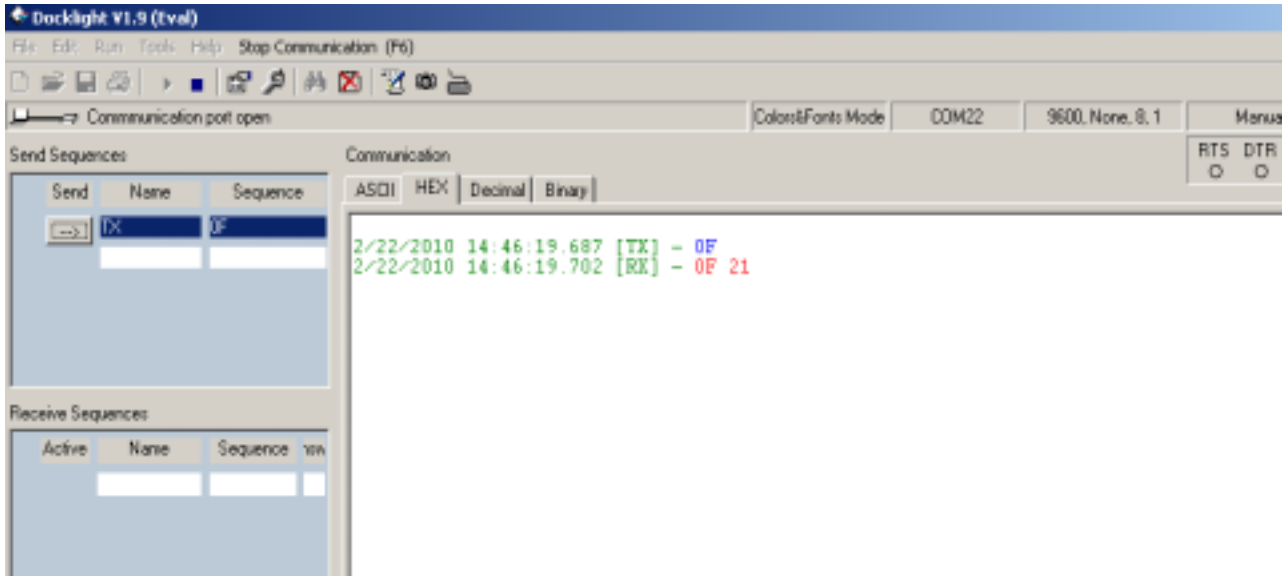

Figure 7 Captured Screenshot of Result

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG