AN2480

AVR315: Using the TWI Module as I2C Master

Introduction

The Two-Wire Serial Interface (TWI) is compatible with Philips I^{2}C protocol. The bus allows simple, robust, and cost-effective communication between integrated circuits in electronics. The strengths of the TWI bus are its capability to address up to 128 devices using the same bus and arbitration, and the possibility to have multiple masters on the bus.

A hardware TWI module is included in most of the Microchip AVR® devices.

This application note describes a TWI master implementation, in the form of a full-featured driver and contains an example of usage for this driver. The driver handles transmission based on the both Standard mode (<100kbps) and Fast mode (<400kbps).

Features

- C-code driver for TWI master
- Compatible with Philips I^{2}C protocol
- Uses the hardware TWI module
- Interrupt driven transmission
- Supports both Standard mode and Fast mode

Table of Contents

1. Overview

This section provides a short description of the TWI interface in general and the TWI module on the AVR 8-bit Microcontroller Family. For more information, refer to the specific device datasheet.

1.1 Two-wire Serial Interface

The Two-wire Serial Interface (TWI) is ideally suited for microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 individually addressable devices using only two bi-directional bus lines; one for clock (SCL) and one for data (SDA). The only external hardware required to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 1-1. TWI Bus Interconnection
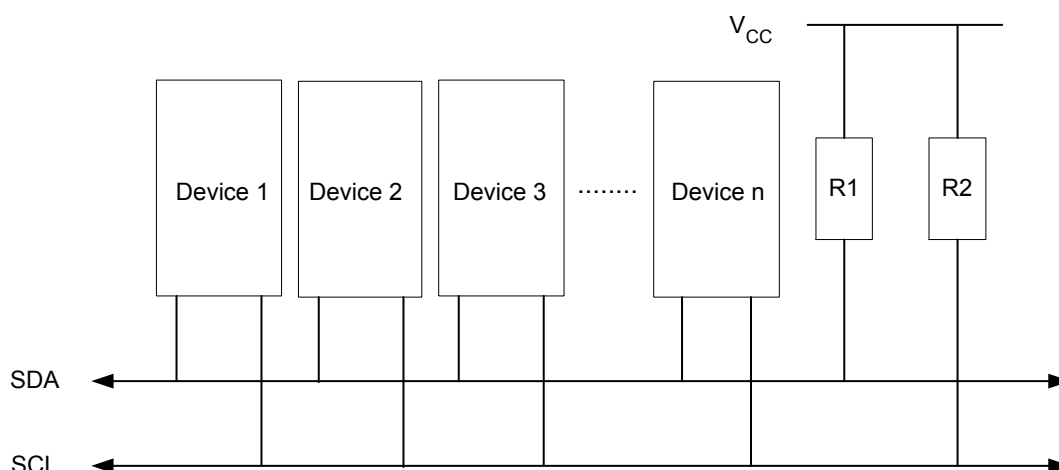
The TWI bus is a multi-master bus where one or more devices are capable of taking control of the bus, can be connected. Only Master devices can drive both the SCL and SDA lines, while a Slave device is only allowed to issue data on the SDA line.

Data transfer is always initiated by a Bus Master device. A high to low transition on the SDA line, while SCL is high, is defined to be a START condition or a repeated start condition.

Figure 1-2. TWI Address and Data Packet Format
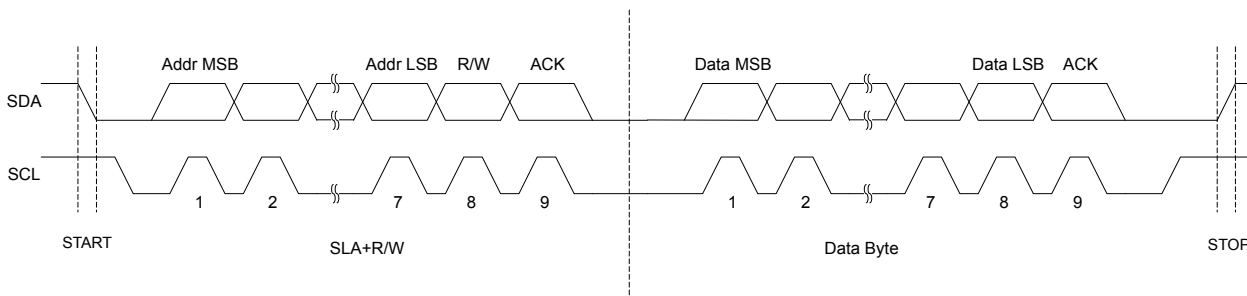
A START condition is always followed by the (unique) 7-bit slave address and then by a Data Direction bit. The Slave device addressed now acknowledges to the Master by holding SDA low for one clock cycle. If the Master does not receive any acknowledge the transfer is terminated. Depending on the Data Direction bit, the Master or Slave now transmits 8-bit of data on the SDA line. The receiving device then acknowledges the data. Multiple bytes can be transferred in one direction before a repeated START or

STOP condition is issued by the Master. The transfer is terminated when the Master issues a STOP condition. A STOP condition is defined by a low to high transition on the SDA line while the SCL is high.

If a Slave device cannot handle incoming data until it has performed some other function, it can hold SCL low to force the Master into a wait-state.

All data packets transmitted on the TWI bus are 9 bits, consisting of one data byte and an acknowledge bit. During a data transfer, the master generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signaled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signaled.

1.2   The AVR TWI Module

The TWI module consists of several sub-modules, as shown in the following figure. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 1-3. Overview of the TWI Module in the AVR Devices
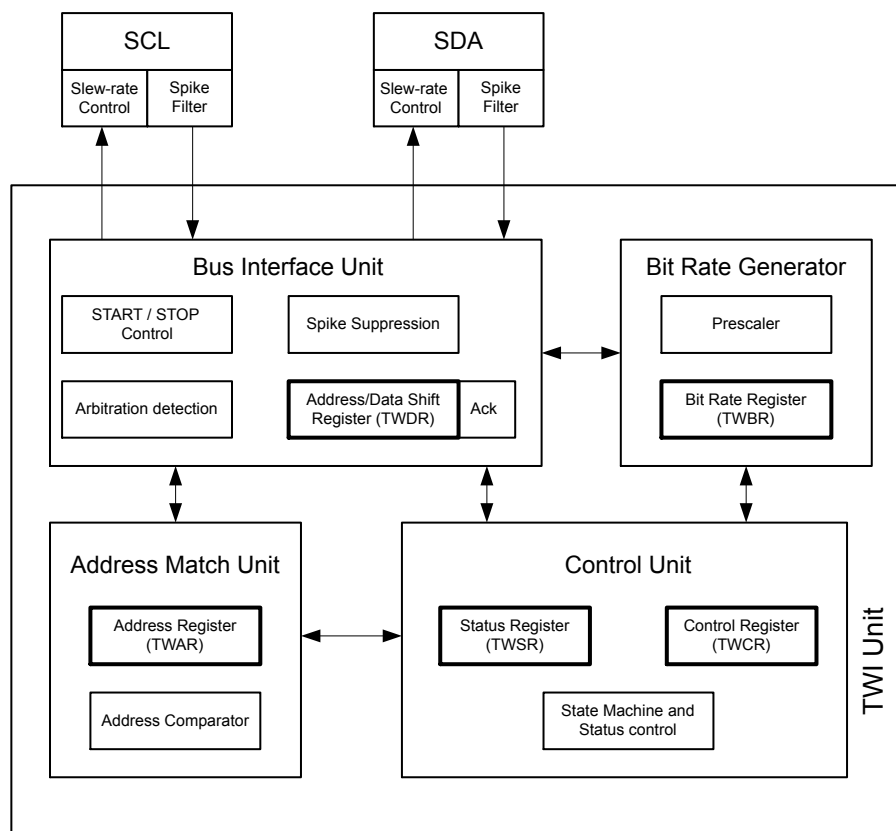
1.2.1   Control Unit

The AVR TWI module can operate in both Master and Slave mode. The mode of operation is distinguished by the TWI status codes in the TWI Status Register (TWSR) and by the use of certain bits in the TWI Control Register (TWCR).

A set of predefined status codes covers the various states the TWI can be present in when a TWI event occurs. The status codes are divided into Master and Slave codes and further in receive and transmit related codes. Status codes for Bus Error and Idle also exist.

The TWI module operates as a state machine and is event driven. If a START CONDITION is followed by a TWI address that matches the address in the Slave's TWI Address Register (TWAR) the TWINT flag is set. This results in the execution of the corresponding interrupt (if Global Interrupt and TWI interrupts are enabled). The firmware of the Slave responds by reading the status code in TWSR and will respond accordingly. All TWI events will set the TWINT flag, and the firmware must respond based on the status in TWSR.

As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set:
- After the TWI has transmitted a START/REPEATED START condition
- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own Slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave
- When a bus error has occurred due to an illegal START or STOP condition

1.2.2 Bit Rate Generator

The Bite Rate Generator unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that Slaves may prolong the SCL low period, thereby reducing the average TWI bus clock frequency. The SCL frequency is generated according to the following equation.

$$SCLfrequency = \frac{CPU\ Clockfrequency}{16 + 2(TWBR) - 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

TWPS is located in TWISR, which is in the same register as where the TWI Status bits are. TWPS is therefore chosen to be set to 0 in this application to simplify the handling of the Status bits, and only TWBR is utilized to achieve the desired speed on SCL. The following table shows a selection of pre-calculated TWBR values based on CPU and SCL frequencies.

Table 1-1. CPU and SCL Frequencies versus Bit Rate Generator Register Settings

CPU Clock Frequency [MHz]	TWBR	TWPS	SCL Frequency [kHz]
16	12	0	400
16	72	0	100
14.4	10	0	400
14.4	64	0	100
12	7	0	400
12	52	0	100
8	2	0	400

| CPU Clock Frequency [MHz]	TWBR	TWPS	SCL Frequency [kHz]
8	32	0	100
4	12	0	100
3.6	10	0	100
2	2	0	100
2	12	0	50
1	2	0	50

1.2.3 SCL and SDA Pins

Both TWI lines (SDA and SCL) are bi-directional, therefore outputs connected to the TWI bus must be of an open-drain or an open-collector type. Each line must be connected to the supply voltage via a pull-up resistor. A line is then logic high when none of the connected devices drives the line, and logic low if one or more drives the line low.

The output drivers contain a slew-rate limiter. The input stages contain a spike suppression unit removing spikes shorter than 50ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. In some systems, the internal pull-ups can eliminate the need for external resistors.

The following figure shows how to connect the TWI units to the TWI bus. The value of Rp depends on VCC and the bus capacitance, typically 4.7kΩ.

Figure 1-4. TWI Connection
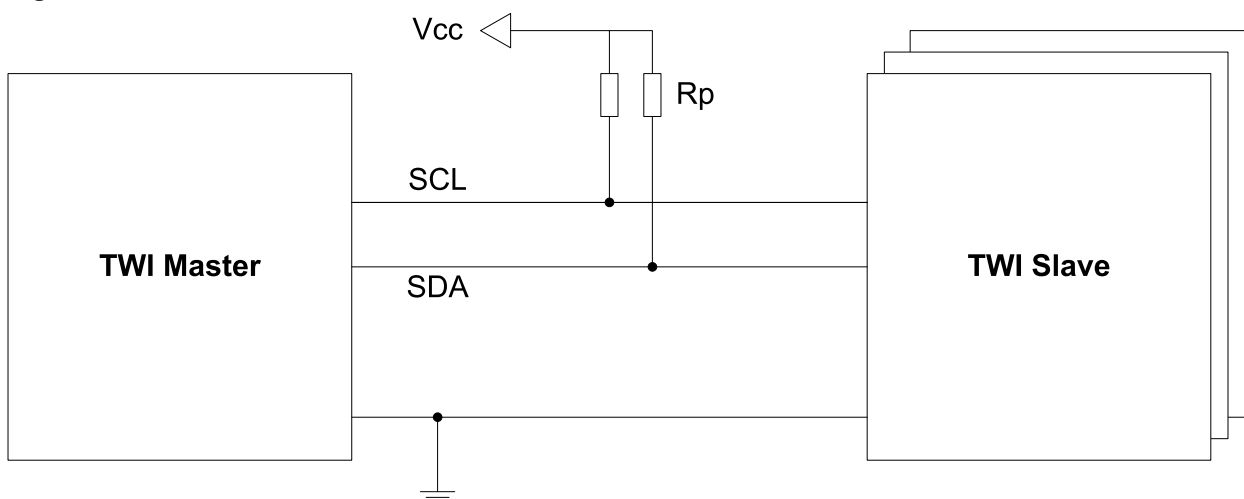
1.2.4 Address Match Unit

The Address Match unit is used only in slave mode, and checks if the received address bytes match the 7-bit address in the TWI Address Register (TWAR). Upon an address match, the Control Unit is informed, allowing the correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR.

Although the clock system to the TWI is turned off in all sleep modes, the interface can still acknowledge its own Slave address or the general call address by using the TWI Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake-up and until the TWINT Flag is cleared.

1.2.5 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller, and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition, it also contains a register containing the (N)ACK bit to be transmitted or received.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master. If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

2. Implementation

The implemented code in this application note is a pure master driver. The TWI modules also support slave operation. See the *AVR311: Using the TWI module as I²C slave* application note for a sample of a slave driver. The master and slave drivers could be merged into one combined master and slave driver, but this is not the scope of this application note.

The master driver C-code consists of three files:

1. `TWI_Master.c`
2. `TWI_Master.h`
3. `Main.c`

There is an example on how to use the driver in the `main.c` file. The `TWI_Master.h` file must be included in the main application and contains all function declarations, a define of the Bit Rate Register (TWBR), and defines for all TWI status codes. The TWBR must be set according to the description earlier in this application note. The TWI status code defines can be used to evaluate error messages and to take appropriate actions. The `TWI_Master.c` file contains all the driver functions.

2.1 Functions

The driver consists of the TWI Interrupt Service Routine and six functions. All functions are available for use outside the driver file scope. However, some of them are also used internally by the driver itself. All functions in the driver are listed in the following table.

Table 2-1. Description of Functions in the TWI Master Driver

Function Name	Description
`void TWI_Master_Initialise( )`	Call this function to set up the TWI master to its initial standby state. Remember to enable interrupts from the main application after initializing the TWI.
`void TWI_Start_Transceiver_with_Data ( unsigned char *msg, unsigned char msgSize)`	Call this function to send a prepared message. The first byte must contain the slave address and the read/write bit. Consecutive bytes contain the data to be sent, or empty locations for data to be read from the slave. Also include how many bytes that should be sent/read including the address byte. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, then initialize the next operation and return.
`void TWI_Start_Transceiver( )`	Call this function to resend the last message. The driver will reuse the data previously put in the transceiver buffers. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, then initialize the next operation and return.
`unsigned char TWI_Transceiver_Busy( )`	Call this function to test if the TWI_ISR is busy transmitting.

Function Name	Description
`unsigned char` `TWI_Get_State_Info( )`	Call this function to fetch the state information of the previous operation. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation. If there was an error, then the function will return the TWIState code.

`unsigned char`
`TWI_Get_Data_From_Transceiver`
`( unsigned char *message,`
`unsigned char messageSize)`
	Call this function to read out the requested data from the TWI transceiver buffer. I.e. first call `TWI_Start_Transceiver` to send a request for data to the slave. Then run this function to collect the data when they have arrived. Include a pointer to where to place the data and the number of bytes requested (including the address field) in the function call. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, before reading out the data and returning. If there was an error in the previous transmission the function will return the TWI error code.

`ISR(TWI_vect)(For GCC)`/
`__interrupt void TWI_ISR(void)`
`(For IAR)`
	This function is the Interrupt Service Routine (ISR), and is automatically called when the TWI interrupt is triggered; that is, whenever a TWI event has occurred. This function should not be called directly from the main application.

The following table consists the description of the driver register byte containing status information from the last transceiver operation. Available as bit fields within a byte.

Table 2-2. Description of the Driver Status Register Byte

`TWI_statusReg`	Description
`TWI_statusReg.lastTransOK`	Set to 1 when an operation has completed successfully.

The following figure shows a flowchart of the process of sending and requesting data over the TWI interface through the drivers. Data is passed through parameters to the functions while the status of an operation is available trough a global status variable.

Figure 2-1. Calling the TWI Driver from the Application
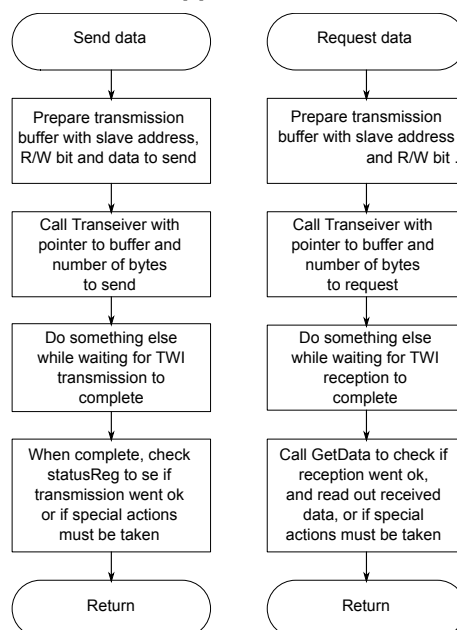
The following figure shows a flowchart, which contains the description of TWI Driver itself . The Transceiver function copies the complete message into the transmission buffer. Then it enables the TWI Interrupt to initiate the transmission. The Interrupt then takes care of the complete transmission and disables itself when the transmission is completed, or if an error state occurs. This way the driver can poll the interrupt enable bit to check if a transmission is ongoing. The main application is only allowed to access the global transceiver variables while the TWI transceiver is not busy. The interrupt stores eventual error states in a variable that is available for the main application through a function call.

Figure 2-2. TWI Driver Functions
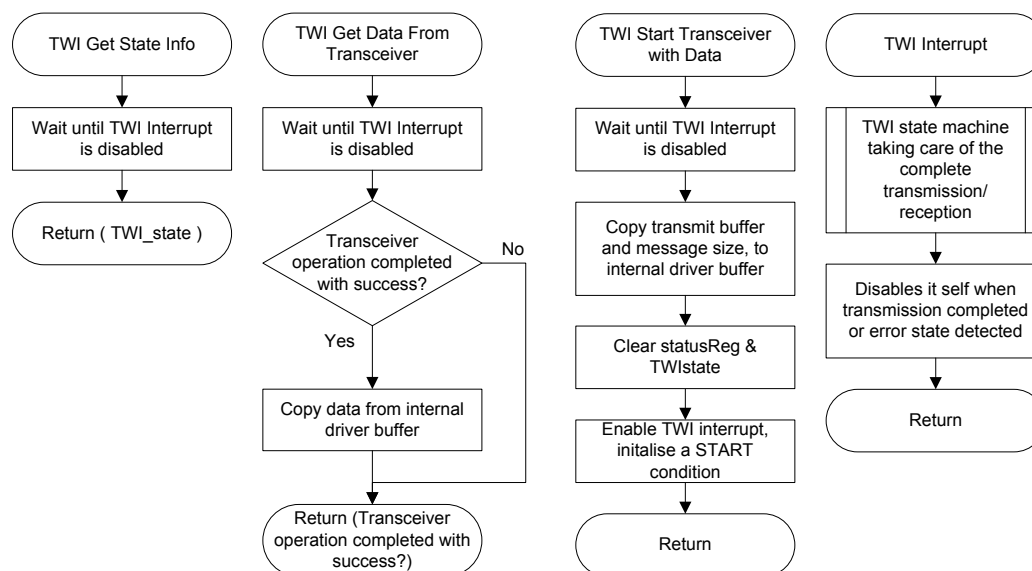
The following flowchart shows a more detailed description of the actions for each event/state in the TWI Interrupt Service Routine. The left column contains the different states/events the TWI state machine can be in when entering the Interrupt. A case switch executes the different actions dependent on the cause of the interrupt call.

Figure 2-3. TWI Interrupt Service Routine

```
TWI Interrupt
    │
    ├──► START has been transmitted ──────────────────────┐
    │                                                       │
    ├──► Repeated START has been transmitted ──────────────┤
    │                                              Reset bufferPointer
    ├──► SLA+W has been transmitted and ACK received ──────┤
    │                                                       │
    ├──► Data byte has been transmitted and ACK received ──┤
    │                                          buffer pointer at the end?
    │                                              No ──┐  Yes ──┐
    ├──► Data byte has been received and ACK transmitted   │      Set Success bit in StatusReg
    │                                                       │           │
    ├──► SLA+R has been transmitted and ACK received ───────┤    Disable TWI interrupt, initalise a STOP
    │                                                       │    condition and clear interrupt flag
    ├──► Data byte has been received and NACK transmitted ──┐
    │                                                       │
    │   Error States                                        │
    │   ┌──────────────────────────────┐                    │
    ├──►│ Arbitration lost             │                    │
    │   │                              │  Initalise a RESTART condition and clear TWI interrupt flag
    ├──►│ SLA+W/R has been transmitted │                    
    │   │ and NACK received            │  Copy from data register to current buffer position.
    │   │                              │
    ├──►│ Data byte has been transmitted│  Set Success bit in StatusReg
    │   │ and NACK received            │
    │   │                              │  Disable TWI interrupt, initalise a STOP
    ├──►│ Bus error due to an illegal  │  condition and clear interrupt flag
    │   │ START or STOP condition      │
    │   └──────────────────────────────┘  Store TWI state information
    │                                              │
    │                                     Disable TWI interrupt
    │
    Copy from data register to current buffer position. Post incr pointer.
    Copy data from current buffer position to data register. Post incr pointer.
    Clear TWI interrupt flag
    buffer pointer position at byte before the end?
        Yes          No
                 Initalise a ACK after reception and clear TWI interrupt flag
    Initalise a NACK after reception and clear TWI interrupt flag
                    Return
```


---

3. Summary

This application note describes the steps to configure the TWI module as a Master and provides an example driver software to implement the module. The firmware for Master Driver is available as a download along with this application note.

4. Revision History

Doc Rev.	Date	Comments
A	07/2017	• Converted to Microchip format and replaced the Atmel® document number 2564 with Microchip DS00002480 • Uploaded new updated images: TWI bus interconnection and TWI address and Data Packet Format • Updated the equation of "Bit rate Generation"
2564D	03/2016	The firmware is ported to Atmel Studio 7
2564C	01/2010	Updated table for bit rates
2564B	09/2007	Removed text about TWBR needs to be higher than 10
2564A	04/2006	Initial document release.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/ support Web Address: www.microchip.com	**Asia Pacific Office** Suites 3707-14, 37th Floor Tower 6, The Gateway Harbour City, Kowloon **Hong Kong** Tel: 852-2943-5100 Fax: 852-2401-3431	**China - Xiamen** Tel: 86-592-2388138 Fax: 86-592-2388130 **China - Zhuhai** Tel: 86-756-3210040 Fax: 86-756-3210049	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829
Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455	**Australia - Sydney** Tel: 61-2-9868-6733 Fax: 61-2-9868-6755 **China - Beijing** Tel: 86-10-8569-7000 Fax: 86-10-8528-2104	**India - Bangalore** Tel: 91-80-3090-4444 Fax: 91-80-3090-4123 **India - New Delhi** Tel: 91-11-4160-8631 Fax: 91-11-4160-8632	**Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79
Austin, TX Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088	**China - Chengdu** Tel: 86-28-8665-5511 Fax: 86-28-8665-7889 **China - Chongqing** Tel: 86-23-8980-9588 Fax: 86-23-8980-9500	**India - Pune** Tel: 91-20-3019-1500 **Japan - Osaka** Tel: 81-6-6152-7160 Fax: 81-6-6152-9310	**France - Saint Cloud** Tel: 33-1-30-60-70-00 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400
Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075	**China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029	**Japan - Tokyo** Tel: 81-3-6880- 3770 Fax: 81-3-6880-3771 **Korea - Daegu** Tel: 82-53-744-4301 Fax: 82-53-744-4302	**Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370
Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924	**China - Hangzhou** Tel: 86-571-8792-8115 Fax: 86-571-8792-8116 **China - Hong Kong SAR** Tel: 852-2943-5100 Fax: 852-2401-3431	**Korea - Seoul** Tel: 82-2-554-7200 Fax: 82-2-558-5932 or 82-2-558-5934	**Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560
Detroit Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983	**China - Nanjing** Tel: 86-25-8473-2460 Fax: 86-25-8473-2470 **China - Qingdao** Tel: 86-532-8502-7355 Fax: 86-532-8502-7205	**Malaysia - Kuala Lumpur** Tel: 60-3-6201-9857 Fax: 60-3-6201-9859 **Malaysia - Penang** Tel: 60-4-227-8870 Fax: 60-4-227-4068	**Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781
Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380	**China - Shanghai** Tel: 86-21-3326-8000 Fax: 86-21-3326-8021 **China - Shenyang** Tel: 86-24-2334-2829 Fax: 86-24-2334-2393	**Philippines - Manila** Tel: 63-2-634-9065 Fax: 63-2-634-9069 **Singapore** Tel: 65-6334-8870 Fax: 65-6334-8850	**Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340
Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800	**China - Shenzhen** Tel: 86-755-8864-2200 Fax: 86-755-8203-1760 **China - Wuhan** Tel: 86-27-5980-5300 Fax: 86-27-5980-5118	**Taiwan - Hsin Chu** Tel: 886-3-5778-366 Fax: 886-3-5770-955 **Taiwan - Kaohsiung** Tel: 886-7-213-7830	**Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50
Raleigh, NC Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270	**China - Xian** Tel: 86-29-8833-7252 Fax: 86-29-8833-7256	**Taiwan - Taipei** Tel: 886-2-2508-8600 Fax: 886-2-2508-0102 **Thailand - Bangkok** Tel: 66-2-694-1351 Fax: 66-2-694-1350	**Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654
Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078			**UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820

Application Note