# Atmel AVR154: Software Emulation of TWI Slave Hardware Module

8-bit Atmel Microcontrollers

Application Note

## Features

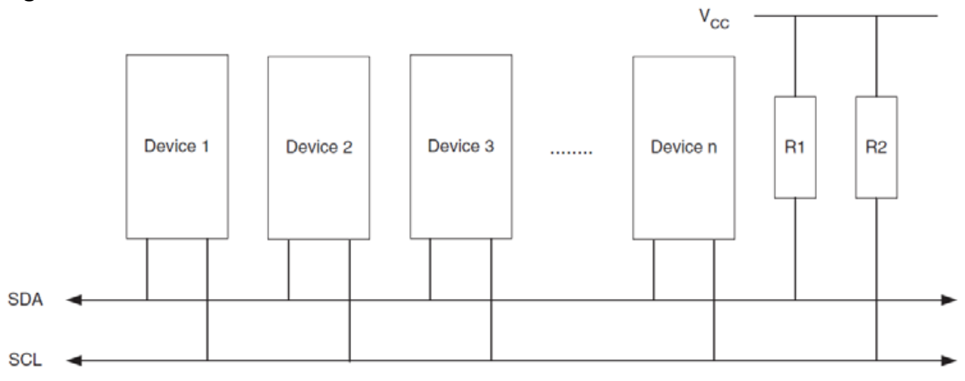
- **C-code bit bang TWI slave driver compatible with Philips' I$^2$C**
- **Completely emulates the hardware TWI slave module**
- **Small code footprint of 713 bytes**
- **Interrupt based implementation**
- **Supports up to 100Kbps**

## 1 Introduction

This application note describes a Two-wire Interface (TWI) slave implementation, in the form of a full-featured driver and an example of usage for this driver. Many chips feature a hardware TWI module; others have Universal Serial Interface (USI), which can be used in TWI mode. This application note discusses the TWI slave driver for devices without any of these hardware modules. This driver emulates the hardware TWI slave in software hence providing the programmer an easy way of porting the code from high end TWI enabled devices to low cost devices.

The Two Wire serial Interface (TWI) is compatible with Philips' I$^2$C protocol. The bus was developed to allow simple, robust, and cost effective communication between integrated circuits in electronics. The strengths of the TWI bus includes the capability of addressing up to 128 devices on the same bus, arbitration, and the possibility to have multiple masters on the bus. Figure 1-1 demonstrates the typical TWI connection in multi slave TWI architecture. This application note does not discuss the theory of TWI interface. More information on TWI protocol can be found in the Atmel® AVR®311 application note.

**Figure 1-1.** TWI Bus interconnection.



Rev. 8478A-AVR-01/12

## 2 Theory

This section gives a short description about the driver architecture and the state machine. It also brings out the comparisons between the hardware TWI module implementation and this driver's software implementation.

### 2.1 Design

The TWI hardware module implementation is driven by an interrupt based on the TWSR register (TWI state register). Since a non-TWI device has no such interrupt available, the only interrupt that is used by this software driver is the falling edge external interrupt on SDA pin.

When the software TWI interface is initialized, this interrupt is enabled and is triggered when a new TWI communication is initiated by the master. This interrupt stays disabled as long as the current transfer is in progress.

A hardware TWI module has TWI states defined for each TWI transaction and the state machine for these states is implemented in hardware. In contrast, in this driver, the TWI states as well as TWI state machine are implemented in software. Once the TWI initialization is detected by the external interrupt, the TWI state machine controls the TWI states and transactions. This state machine is equivalent to the TWI interrupt service routine in the hardware TWI module implementation.

## 3 Prerequisites

This document requires basic familiarity with following:

- Compiling C projects with IAR Embedded Workbench® for Atmel AVR, as this driver is written using this IDE
- General familiarity with TWI interfaces and electrical connection requirements
- A method to debug and test the compiled application, or download the application hex files into the targeted device, such as the Atmel AVR JTAGICE mkII or Atmel AVR JTAGICE 3

## 4 Limitations

- This driver requires a device running at minimum 4MHz system clock for 100KHz TWI operation. This code was written and tested on Atmel ATMega16HVA device running at 4MHz. 100KHz rate is not warranted below 4MHz system clock
- This driver uses two GPIO pins of the device. One of these pins must have external interrupt triggering capability. This driver uses negative edge external interrupt on the pin used as SDA
- Since it is a software TWI driver, the driver is not able to detect bus contention errors like the hardware TWI module
- This driver uses three dedicated general purpose registers R13-R15 to emulate hardware TWI module registers TWSR, TWDR, and TWEA. The user needs to specifically set them in the IAR™ compiler settings under Project Options->C/C++ compiler->Code->Register Utilization. Use of dedicated registers increases the code efficiency and speed

NOTE

If the user does not want to use dedicated registers for TWSR, TWDR and TWEA, he must define these as global variables and delete their declarations in the TWI.h file.

## 5 Creating a project

The driver consists of only two files:

1. TWI.c
2. TWI.h

TWI.c contains the function definitions, incoming/outgoing buffers, TWI state machine implementation and the interrupt service routine. The size of incoming and outgoing buffers can be set by the user.

TWI.h contains the function declarations as well as settable parameters for the TWI interface. These parameters are listed below:

```
#define SDA PC0
#define SCL PA0
#define SLAVE_ADDRESS 0x5D
```

SDA is the GPIO pin to be used as SDA. In this example, it is PC0.

SCL is the GPIO pin to be used as SCL. In this example, it is PA0.

SLAVE_ADDRESS is the 7-bit slave address for this slave. In this example it is 0x5D

To use this driver, both these files should be included in the project and TWI.h should be included in the main source file.

## 6 Using the driver

The driver code contains a demo application file named ‘main.c’ which demonstrates the initialization and use of this driver. Since it is an interrupt based driver, global interrupt must be enabled before using this driver. After enabling global interrupt, call the functions below to start the TWI slave interface.

```
twi_slave_init();
twi_slave_enable();
```

Both these functions must to be executed to turn the TWI slave module on.

The software uses three general purpose registers R13, R14, and R15 as dedicated TWI registers. These registers are used just like hardware TWI registers TWEA, TWDR and TWSR. The TWI state machine uses these registers to synchronize and carry out TWI transactions with the master and updates the incoming and outgoing buffer.

With every write request from the master, the incoming buffer gets updated with the latest received data. It is up to the user to use the data and clear the buffer. Similarly, with every read request from the master, the data in the outgoing buffer is sent out to the master. It is up to the user to populate the outgoing buffer. The length of data transaction is only limited by the size of incoming and outgoing buffers.

## 7 References

AVR TWI for beginners:
www.atmel.com/dyn/resources/prod_documents/doc2564.pdf
IAR Embedded Workbench:
http://www.iar.com/en/Products/IAR-Embedded-Workbench/AVR/

**Atmel Corporation**
2325 Orchard Parkway
San Jose, CA 95131
USA
**Tel:** (+1)(408) 441-0311
**Fax:** (+1)(408) 487-2600
www.atmel.com

**Atmel Asia Limited**
Unit 01-5 & 16, 19F
BEA Tower, Milennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
**Tel:** (+852) 2245-6100
**Fax:** (+852) 2722-1369

**Atmel Munich GmbH**
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
**Tel:** (+49) 89-31970-0
**Fax:** (+49) 89-3194621

**Atmel Japan**
16F, Shin Osaki Kangyo Bldg.
1-6-4 Osaki Shinagawa-ku
Tokyo 104-0032
JAPAN
**Tel:** (+81) 3-6417-0300
**Fax:** (+81) 3-6417-0370