ATWILC1000

ATWILC1000 RTOS Driver Porting Guide

Introduction

This porting guide describes how to integrate the ATWILC1000 WLAN module to communicate with any MCU via the Serial Peripheral Interface (SPI) or the Secure Digital Input Output (SDIO) interfaces. This guide also provides an example for interfacing the SPI and SDIO with the MCU.

Prerequisites

- Hardware Prerequisites:
  - SAM4S Xplained evaluation kit
  - ATWILC1000 board
  - Micro-USB cable (TypeA / MicroB)
- Software Prerequisites:
  - Atmel Studio 6
  - ATWILC1000 software release package

Table of Contents

1. SPI Interface

The ATWILC1000 external interfaces include:

- I^2C Slave interface controls and configures using PC GUI tool
- SPI Slave and SDIO Slave interfaces controls and transfers data

This porting guide focuses on the SPI that operates as a SPI Slave.
Note: For more information on interfacing sample with the ATWILC1000 and SAM4S Xplained board, refer to Appendix.
For more information on the ATWILC1000, refer to the *ATWILC1000-MR110xB Datasheet*.

The ATWILC1000 provides a Serial Peripheral Interface (SPI) that can be used to control and transfer of serial I/O of 802.11 data. The SPI Slave pins are mapped as shown in the following table. The RXD pin is same as the Master Output Slave Input (MOSI), and the TXD pin is same as the Master Input Slave Output (MISO). The SPI Slave is a full-duplex slave synchronous serial interface that is available after reset, when pin 9 (SDIO_SPI_CFG) is connected to VDDIO.

Table 1-1. ATWILC1000 SPI Slave Interface Pin Mapping

Pin No.	SPI Function
9	CFG - Must be connected with VDDIO
16	SSN - Active Low Slave Select
18	SCK - Serial Clock
13	RXD - Serial Data Receive
17	TXD - Serial Data Transmit

When the SPI is not selected, that is, when SSN is high, the SPI does not interfere with data transfers between the serial-master and other serial-slave devices. When the serial-slave is not selected, its transmitted data output is buffered, resulting in a high-impedance driver on the serial master receive line. The SPI Slave interface responds to a protocol that allows an external host to read or write any register in the chipset and also initiates DMA transfer.

The SPI Slave interface supports four standard modes as determined by the Clock Polarity (CPOL) and Clock Phase (CPHA) settings. These modes are illustrated in the following figure and table.

Figure 1-1. ATWILC1000 SPI Host MCU Interface Diagram

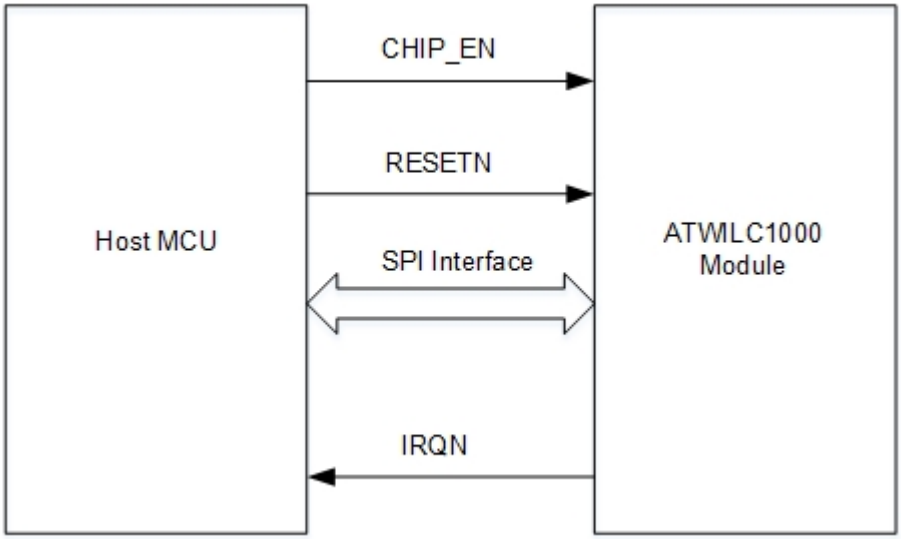
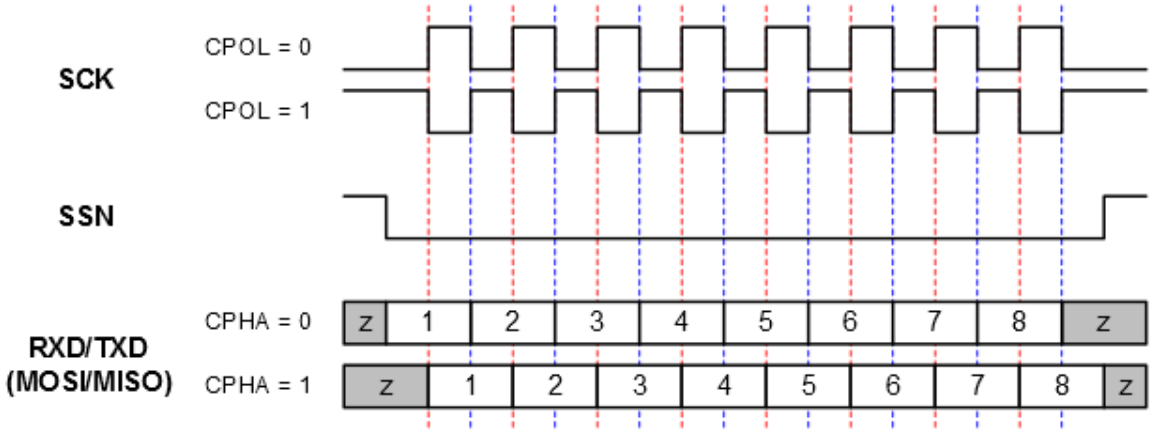


Table 1-2. SPI Slave Modes

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

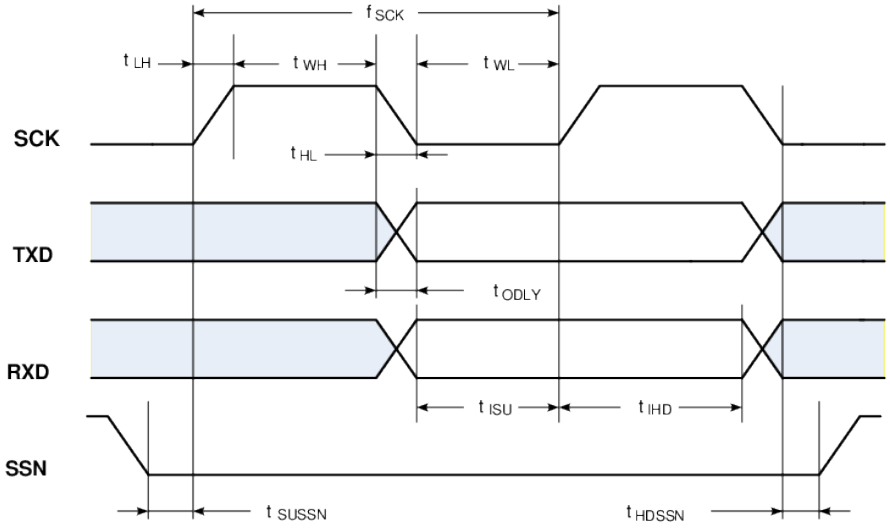
The red lines in the following figure correspond to Clock Phase at 0 and the blue lines correspond to Clock Phase at 1.

Figure 1-2. SPI Slave Clock Polarity and Clock Phase Timing



The following figure shows the ATWILC1000 SPI timing diagram.

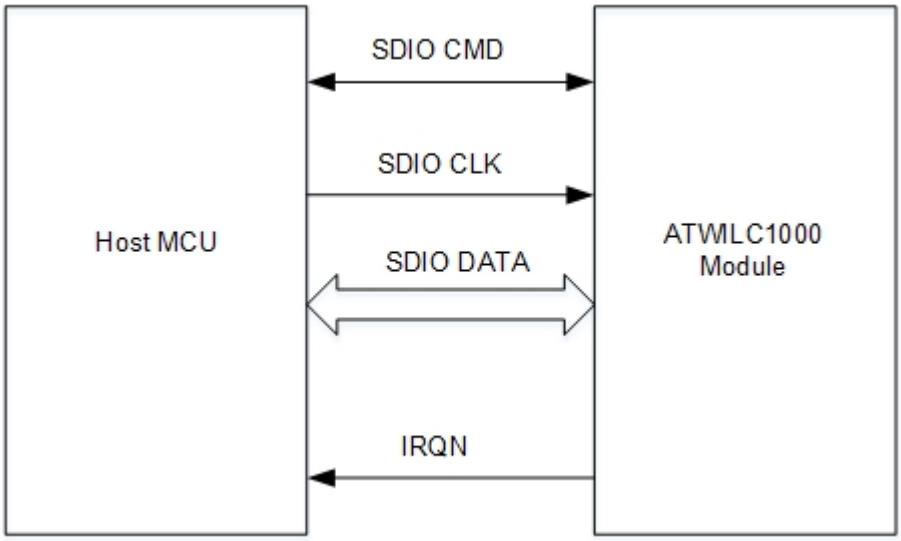
Figure 1-3. ATWILC1000 SPI Timing Diagram



2. SDIO Interface with ATWILC1000

The following figure shows the SDIO interface connection with the Host MCU and ATWILC1000.

Figure 2-1. ATWILC1000 SDIO Host MCU Interface Diagram



For more details, refer to the External Interfaces chapter in the *ATWILC1000-MR110xB Datasheet*.

3. ATWILC1000 Host Interface Driver

This section describes the communication between the ATWILC1000 host interface layer with a host MCU via the serial interfaces supported by the ATWILC1000 software.

The ATWILC1000 software architecture consists of the following three components:

- IoT Application layer
- ATWILC1000 firmware
- ATWILC1000 ASIC driver

The ATWILC1000 ASIC driver includes the host interface layer to communicate with the host MCUs, for example, to communicate with the SAM4S via the SPI or SDIO interface.

3.1 Host Driver Configuration for SPI Interface

To communicate with a host application CPU, define the `CONF_WILC_USE_SPI` macro for the SPI interface. For more information on the SPI features, refer to the `CONF_WILC_USE_SPI` macro in the driver source codes. This macro can be defined in the Atmel Studio compiler symbols.

3.2 Host Driver Configuration for SDIO Interface

To communicate with a host application CPU, define the `CONF_WILC_USE_SDIO` and `CONF_WILC_USE_SDIO_EXT_IRQ`macros for the SDIO interface and external IRQ. For more information on the SDIO features, refer to the `CONF_WILC_USE_SDIO` macro in the driver source codes. This macro can be defined in the Atmel Studio compiler symbols.

3.3 Platform Dependent Driver

The ATWILC1000 driver consists of platform independent and dependent parts. The dependent part of the ATWILC1000 driver must be ported to specific platforms. The ATWILC1000 driver defines the functions that must be ported to communicate with a host CPU via the SPI interface. The Bus Wrapper APIs lists the declarations defined in the `nm_bus_wrapper.h` file, which provides wrapper functions to a specific platform.

Note: The `nm_bus_wrapper.h` file is located in the `\src\ASF\common\components\wifi\wilc\bus_wrapper\` folder.

3.4 Bus Wrapper APIs

This section describes the host bus wrapper functions. The independent parts of the ATWILC1000 driver calls the functions in a specified time. For example, the `nm_bus_init` function calls during driver initial stage. Similarly, the `nm_bus_deinit` function is called when de-initializing the driver. The `nm_bus_ioctl` function is called when the driver reads or writes the control packets or 802.11 data.

```
nm_bus_init
    Declaration
        sint8 nm_bus_init (void *)
    Description
        Initialize the bus wrapper
    Return
        M2M_SUCCESS in case of success and M2M_ERR_BUS_FAIL in case of failure
```

```
    Declaration
        sint8 nm_bus_deinit (void)
    Description
        De-initialize the bus wrapper
    Return
        ZERO in case of success and M2M_ERR_BUS_FAIL in case of failure

nm_bus_ioctl
    Declaration
        sint8 nm_bus_ioctl (uint8 u8Cmd, void* pvParameter)
    Description
        send/receive from the bus
    Parameters
        U8Cmd: IOCTL command for the operation
        pvParameter: Arbitrary parameter depending on IOCTL
    Return
        M2M_SUCCESS in case of success and M2M_ERR_BUS_FAIL in case of failure
```

For more details on how to port the bus wrapper functions on the Xplained board, refer to the Porting on SAM4S Xplained Board chapter.

4. Porting on SAM4S Xplained Board

This chapter describes how to port the ATWILC1000 SPI bus wrapper with the SAM4S Xplained board. For more details on the SAM4S, refer to the SAM4S Xplained Pro User Guide and SAM4S MCUs for SAM4S MCUs. The Appendix shows an example from Atmel Studio for reading the chip ID with the ATWILC1000. This example can be used to verify porting of the SPI slave bus wrapper.

4.1 Writing `nm_bus_wrapper.c` File

The ATWILC1000 release package includes the `nm_bus_wrapper.h` file to declare the functions that are required for the driver, which is a dependent part of the ATWILC1000 ASIC driver. Defining SPI Wrapper Functions describes how to write the bus wrapper C-file to define and declare in the `nm_bus_wrapper.h` file. It is required to create a new file to define declarations, for example, the `nm_bus_wrapper_sam4s.c` and `sdio_sam4s.c` files indicate that the file is for the SAM4S board.
Note: Make sure to add a new file in the compile list.

4.2 Defining SPI Wrapper Functions

The following sample code shows how to initialize the SPI driver to provide the wrapper function `nm_bus_init` to the ATWILC1000 ASIC driver. This function initializes the SPI driver of the SAM4S. The ATWILC1000 ASIC driver requires the host interface layer to communicate with a host MCU, which calls this function during the ATWILC1000 initial phase. Then, the ATWILC1000 ASIC driver reads the chip ID to validate the SPI communication.

```
sint8 nm_bus_init(void *pvInitValue)
{
    sint8 result = M2M_SUCCESS;
#ifdef defined CONF_WILC_USE_SPI

    /* Configure SPI pins. */
    gpio_configure_pin(CONF_WILC_SPI_MISO_GPIO,
                       CONF_WILC_SPI_MISO_FLAGS);
    gpio_configure_pin(CONF_WILC_SPI_MOSI_GPIO,
                       CONF_WILC_SPI_MOSI_FLAGS);
    gpio_configure_pin(CONF_WILC_SPI_CLK_GPIO,
                       CONF_WILC_SPI_CLK_FLAGS);
    gpio_configure_pin(CONF_WILC_SPI_CS_GPIO,
                       CONF_WILC_SPI_CS_FLAGS);

    /* Get the PIO instance used for CS. */
    p_pio_cs = (Pio *)((uint32_t)PIOA + (PIO_DELTA *
                       (CONF_WILC_SPI_CS_GPIO >> 5)));
    SPI_DEASSERT_CS();

    /* Configure SPI module. */
    spi_enable_clock(CONF_WILC_SPI);
    spi_disable(CONF_WILC_SPI);
    spi_reset(CONF_WILC_SPI);
    spi_set_master_mode(CONF_WILC_SPI);
    spi_disable_mode_fault_detect(CONF_WILC_SPI);
#ifdef    __SAM4SD32C__
    spi_set_fixed_peripheral_select(CONF_WILC_SPI);
#else
    spi_set_peripheral_chip_select_value(CONF_WILC_SPI,
                                         CONF_WILC_SPI_NPCS);
#endif
    spi_set_clock_polarity(CONF_WILC_SPI,
                           CONF_WILC_SPI_NPCS, CONF_WILC_SPI_POL);
    spi_set_clock_phase(CONF_WILC_SPI,
                           CONF_WILC_SPI_NPCS, CONF_WILC_SPI_PHA);
    spi_set_bits_per_transfer(CONF_WILC_SPI,
```

```
                            CONF_WILC_SPI_NPCS, SPI_CSR_BITS_8_BIT);
    if (sysclk_get_cpu_hz() % CONF_WILC_SPI_CLOCK != 0)
        {
            M2M_ERR("Warning: non-integer SPI clock divider not allowed, was floored to %d.\r
\n",
                             sysclk_get_cpu_hz()/ CONF_WILC_SPI_CLOCK);
        }
    spi_set_baudrate_div(CONF_WILC_SPI, CONF_WILC_SPI_NPCS,
                            (sysclk_get_cpu_hz() / CONF_WILC_SPI_CLOCK));
    spi_set_transfer_delay(CONF_WILC_SPI, CONF_WILC_SPI_NPCS,
                           CONF_WILC_SPI_DLYBS,CONF_WILC_SPI_DLYBCT);
    spi_enable(CONF_WILC_SPI);

    /* Get pointer to SPI master PDC register base. */
    g_p_pdc_spi = spi_get_pdc_base(CONF_WILC_SPI);
    pdc_disable_transfer(g_p_pdc_spi, PERIPH_PTCR_RXTDIS | PERIPH_PTCR_TXTDIS);
    nm_bsp_reset();
    SPI_DEASSERT_CS();
#elif defined CONF_WILC_USE_SDIO
    result = sam4s_sdio_init();
#endif
    return result;
}
```

The `nm_bus_ioctl` function is called when the ATWILC1000 ASIC driver reads or writes the data. In case of the SPI interface, only the `NM_BUS_IOCTL_RW` command is used.

```
sint8 nm_bus_ioctl(uint8 u8Cmd, void* pvParameter)
    {
        sint8 s8Ret = 0;
        switch(u8Cmd)
        {
    #ifdef defined CONF_WILC_USE_SPI
                case NM_BUS_IOCTL_RW:
                {
                tstrNmSpiRw *pstrParam = (tstrNmSpiRw *)pvParameter;
                s8Ret = spi_rw(pstrParam->pu8InBuf,
                pstrParam->pu8OutBuf, pstrParam->u16Sz);
                }
            break;
        }
    return s8Ret;
    }
```

The wrapper function also provides the `nm_bus_deinit` function. If required in a specific platform, the function must also be implemented in the wrapper file. If not required, return zero as shown in the following sample code. This function is common for the SPI and SDIO interfaces.

```
sint8 nm_bus_deinit(void)
    {
        return M2M_SUCCESS;
    }
```

4.3 Defining SDIO Wrapper Functions

The SDIO interface initialization is implemented in the `\src\ASF\common\components\wifi\wilc\bus_wrapper\source\sdio_sam4s.c` file. To initialize the SDIO interface, define the `CONF_BOARD_SD_MMC_HSMCI` marco in the `\src\config\conf_board.h` file. This board- dependent implementation is based on the board support package.

```
int8_t sam4s_sdio_init(void)
{
    #if defined (CONF_BOARD_SD_MMC_HSMCI)
    /* Configure HSMCI pins */
    gpio_configure_pin(PIN_HSMCI_MCCDA_GPIO,
                       PIN_HSMCI_MCCDA_FLAGS);
```

```
    gpio_configure_pin(PIN_HSMCI_MCCK_GPIO,
                       PIN_HSMCI_MCCK_FLAGS);
    gpio_configure_pin(PIN_HSMCI_MCDA0_GPIO,
                       PIN_HSMCI_MCDA0_FLAGS);
    gpio_configure_pin(PIN_HSMCI_MCDA1_GPIO,
                       PIN_HSMCI_MCDA1_FLAGS);
    gpio_configure_pin(PIN_HSMCI_MCDA2_GPIO,
                       PIN_HSMCI_MCDA2_FLAGS);
    gpio_configure_pin(PIN_HSMCI_MCDA3_GPIO,
                       PIN_HSMCI_MCDA3_FLAGS);

    /* Configure SD/MMC card detect pin */
    #ifdef SD_MMC_0_CD_GPIO
        gpio_configure_pin(SD_MMC_0_CD_GPIO, SD_MMC_0_CD_FLAGS);
    #endif
        #endif

        // Initialize SDIO on SAM4s
        sd_mmc_init();

        //Wait for a card and ready
        check_card_exist();


        // Display basic card information
        main_display_info_card(0);

        return M2M_SUCCESS;
}
```

The `nm_bus_ioctl` function is called when the ATWILC1000 ASIC driver reads or writes the data. For the SDIO interface, `NM_BUS_IOCTL_CMD_52` and `NM_BUS_IOCTL_CMD_53` commands are used.

```
sint8 nm_bus_ioctl(uint8 u8Cmd, void* pvParameter)
    {
        sint8 s8Ret =  0;
        switch(u8Cmd)
        {
    #ifdef defined CONF_WILC_USE_SDIO
            case NM_BUS_IOCTL_CMD_52:
            {
            tstrNmSdioCmd52* pstrParam = (tstrNmSdioCmd52 *)pvParameter;
            s8Ret = nmi_sdio_cmd52(pstrParam);
            }
            break;
            case NM_BUS_IOCTL_CMD_53:
            {
                tstrNmSdioCmd53* pstrParam = (tstrNmSdioCmd53 *)pvParameter;
                s8Ret = nmi_sdio_cmd53(pstrParam);
            }
            break;
    #endif
        }
        return s8Ret;
    }
```

Note: To handle the SDIO IRQ, the IRQN pin of the ATWILC1000 module must be connected with the host MCU external interrupt pin. In this driver, the ATWILC1000 does not support the in-band interrupt option.

5. Reference Documentation

The following table provides the set of collateral documents to ease integration and device ramp.

Table 5-1. Reference Documents

Title	Content
ATWILC1000 RTOS Driver Porting Guide	This document.
ATWILC1000-MR110xB Datasheet	This document describes the functional description of IEEE® 802.11 b/g/n Link Controller module.

For a complete listing of development support tools and documentation, visit http://www.microchip.com/, or contact the nearest Microchip field representative.

6. Document Revision History

Rev A - 1/2018

Section	Changes
Document	Initial release

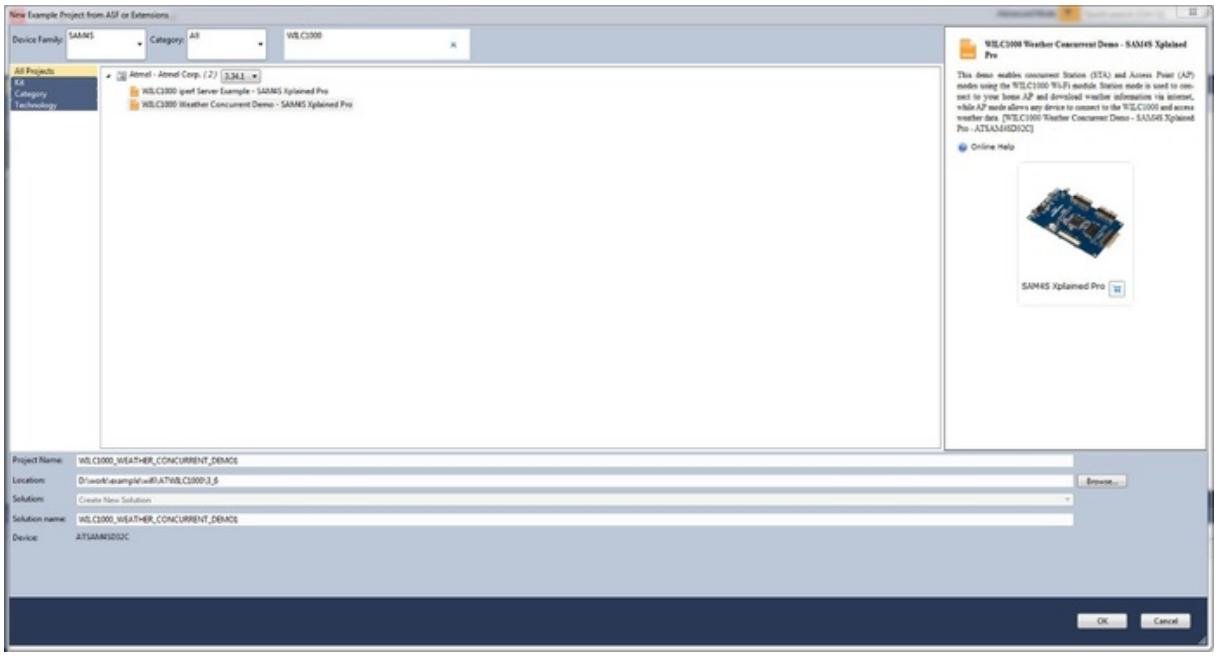
7. Appendix - Get ATWILC1000 Examples

The Atmel Studio is required in order to work on the ATWILC1000 with the SAM4S board. This appendix shows how to get to the ATWILC1000 examples from the Atmel Studio, and download and install the Atmel Studio. Refer to the Atmel Studio online help for more instructions.

Perform the following to find the ATWILC1000 examples in the Atmel Studio.

- Launch the Atmel Studio
- Install the latest ASF release or select the ATWILC1000 project from the release package

Figure 7-1. New Example Project from ASF or Extension



This installation creates a new project in the specified location, for an example to read the chip ID. If the SPI Slave interface is not working, it fails to read the chip ID. For more information on this example, refer to the Atmel gallery.

This example contains the `nm_bus_wrapper_SAM4S.c` file in the `/src/winc/bus_wrapper/source` location. This file ports the SPI Slave interface driver for the ATWILC1000 ASIC driver. This file contains three implementations for the `nm_bus_init`, `nm_bus_ioctl` and `nm_bus_deinit` as explained in this guide.

The main function contains a simple implementation. After calling the `m2m_wifi_init` function, it prints the chip ID to check functioning of the bus interface.

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

Trademarks

ISBN: 978-1-5224-2577-9

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 **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **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 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820