MICROCHIP

AN3664

AVR® DA Low-Power Features and Sleep Modes

Introduction

Author: Ioan Pop, Microchip Technology Inc.

This document presents an overview of the low-power modes and features of the AVR® DA microcontroller family. The current consumption in the different sleep modes is compared with the help of the Power Debugger (ATPOWERDEBUGGER).

Furthermore, this paper explains how to modify the Curiosity Nano board to perform power readings, how to use the Power Debugger, and describes the available sleep modes and other power-saving features.

View Code Used for Testing on GitHub
Click to browse repositories

Table of Contents

1. Sleep Modes and the Sleep Controller

The AVR DA microcontrollers can operate in four modes: Active, Idle, Standby, and Power-Down. The Active mode is the mode where code can be executed. The other three modes are sleep modes that reduce the overall power consumption of the application.

These modes are set through the Sleep Mode (SMODE) bit field in the Control A (SMODE.CTRLA) register of the Sleep Controller. The Sleep Enable (SEN) bit in the same register needs to be set for activating one of the sleep modes.

For temperature ranges above 70°C, the HTLLEN bit in the SLPCTRL.VREGCTRL register can be used to reduce power consumption (High-Temperature Low-Leakage Enable). When this is enabled, the address match and the Configurable Custom Logic (CCL) wake-up sources are unusable and must be disabled by the user.

Figure 1-1. Control A Register for Sleep Controller

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					SMODE[2:0]			SEN
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:1 – SMODE[2:0] Sleep Mode
Writing these bits selects the desired sleep mode when the Sleep Enable (SEN) bit is written to '`1`' and the `SLEEP` instruction is executed.

Value	Name	Description
`0x0`	IDLE	Idle mode enabled
`0x1`	STANDBY	Standby mode enabled
`0x2`	PDOWN	Power-Down mode enabled
`Other`	-	Reserved

Bit 0 – SEN Sleep Enable
This bit must be written to '`1`' before the `SLEEP` instruction is executed to make the microcontroller enter the selected sleep mode.

1.1 Idle Sleep Mode

In Idle sleep mode, the code execution is stopped, all peripherals are running, and all interrupt sources can wake up the devices from sleep.

1.2 Standby Sleep Mode

In Standby sleep mode, the power consumption is dependent on which peripherals are enabled. Most peripherals have a Run-In Standby (RUNSTBY) bit that can be set to allow the peripheral to run while in this sleep mode.

1.3 Power-Down Sleep Mode

In Power-Down sleep mode, all peripherals are stopped, except for the Watchdog Timer (WDT) and the Periodic Interrupt Timer (PIT). The only wake-up sources are the pin change interrupt, the TWI address match, and the CCL with filter and edge-detect disabled.

In this mode, the High-Frequency Oscillator (OSCHF) is stopped, and only the 32.768 kHz oscillator is running.

1.4 `avr/sleep.h` Library

The assembly instruction that tells the microcontroller to enter sleep is the `SLEEP` instruction. For easier integration with C code, the `avr/sleep.h` library was created to offer functions to facilitate entering sleep.

The C library `avr/sleep.h` provides the following functions: `sleep_enable()`, `sleep_disable()`, `set_sleep_mode(<mode>)`, `sleep_cpu()` and `sleep_mode()`.

The `set_sleep_mode(<mode>)` function takes as argument one of the following macros: `SLEEP_MODE_IDLE`, `SLEEP_MODE_STANDBY` or `SLEEP_MODE_PWR_DOWN`. Using this function with one of the macros will set the corresponding sleep mode.

The `sleep_enable()` and `sleep_disable()` functions set and respectively clear the SEN bit.

The `sleep_cpu()` function invokes the `SLEEP` instruction.

The `sleep_mode()` function combines the previous functions. It calls `sleep_enable()`, then `sleep_cpu()`. When the device wakes up, the `sleep_disable()` function is called and then the program continues as usual.

1.5 Entering a Sleep Mode

To enter sleep, first set the required sleep mode with the `set_sleep_mode(<mode>)` function and then call `sleep_mode()`. The microcontroller is now in sleep and will wake up when the correct interrupt is received.

1.6 Exiting a Sleep Mode

The microcontroller wakes up from sleep when it receives an interrupt. This implies that the global interrupts must be enabled to wake up from sleep. Depending on the sleep mode, only some sources will wake up the device.

In Idle sleep mode, any enabled interrupt source will wake up the device.

In Standby sleep mode, only the peripherals configured to run in Standby sleep mode can generate interrupts.

In Power-Down sleep mode, only the interrupts from the PIT, Watchdog, pin changes, TWI address match, and CCL with filter and edge-detect disabled can wake up the device.

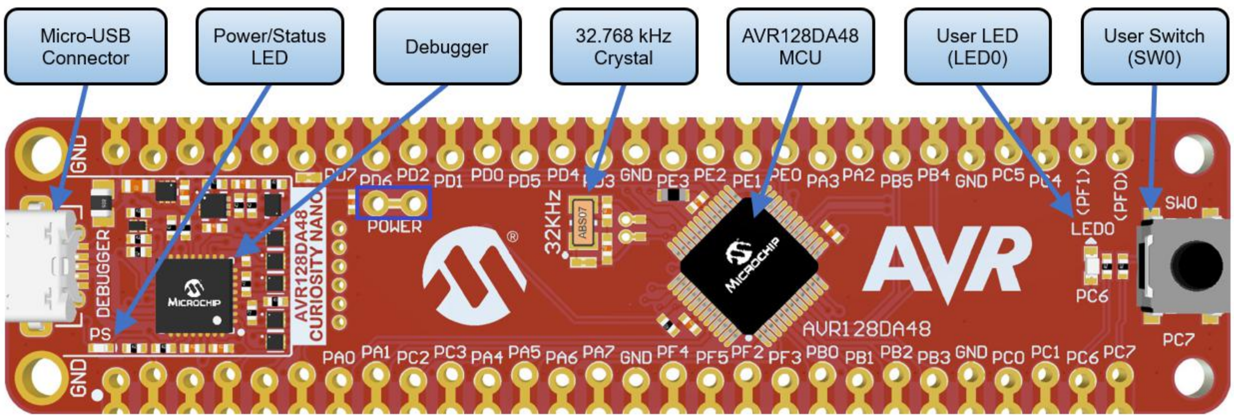
It takes six clock cycles to wake up from sleep plus the time necessary for the main oscillator and regulator to start when waking up from Standby or Power-Down sleep mode.

2. Modifying the Curiosity Nano Board

The AVR128DA48 Curiosity Nano board comes equipped with the microcontroller mentioned above, a regulator, and a Nano Embedded Debugger. These devices are helpful for programming quickly through the USB port, but they introduce power consumption, and accurate power readings cannot be obtained.

Fortunately, the Curiosity Nano board offers two holes for pins. The holes are connected with a copper trace labeled POWER. If this trace is cut off and pins are soldered in the two holes, an ammeter can be connected at these points that will provide accurate consumption for the device.

Figure 2-1. Curiosity Nano Board



The current flows from the hole closest to the USB connector to the one closest to the microcontroller.

Cutting the trace means that to program the microcontroller, the two soldered pins need to be connected by a jumper when in normal use.

Figure 2-2. Modified Curiosity Nano Board

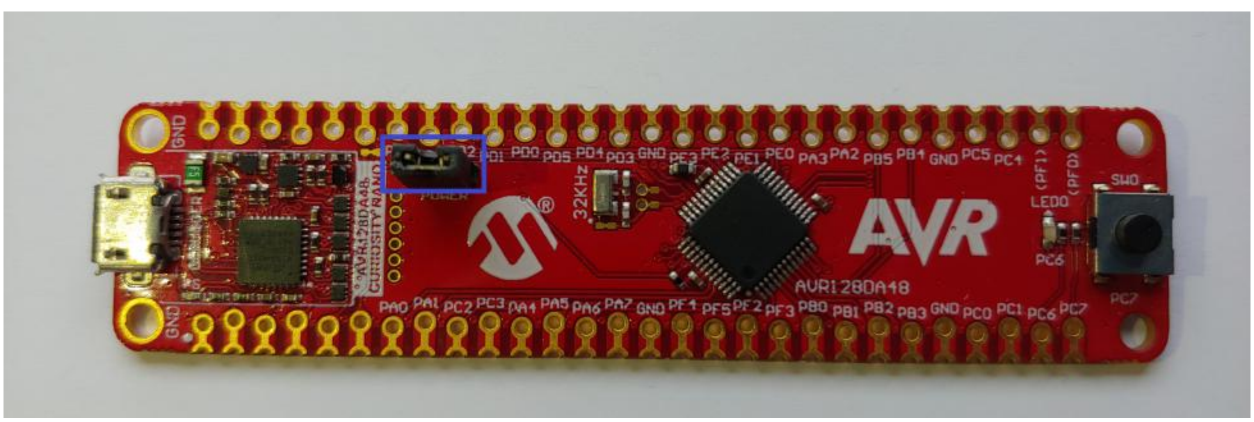


Figure 2-2 shows a modified board where two pins have been connected by a jumper to allow programming.

3. Using the Power Debugger for Power Readings

Info: These power readings can be taken with a multimeter, in ammeter mode, connected between the two soldered pins of the Curiosity Nano board.

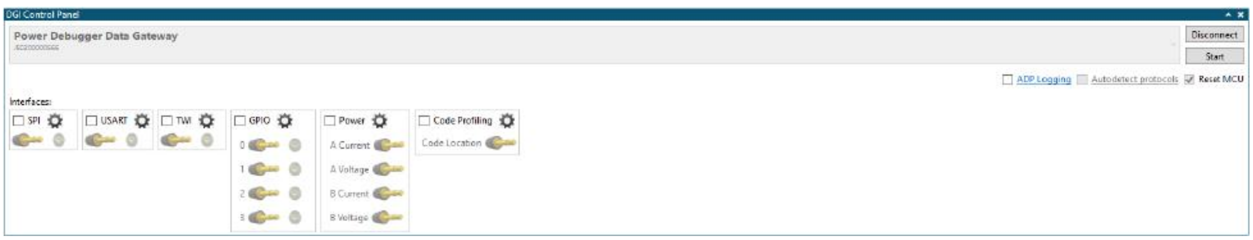
Figure 3-1. Current Reading with Multimeter Setup

To use the Power Debugger, Atmel Studio's Data visualizer needs to be utilized. Open Atmel Studio, go to *Tools → Data Visualizer*. Expand the **DGI Control Panel** tab and select the **Power Debugger Data Gateway**.

Figure 3-2. DGI Control Panel

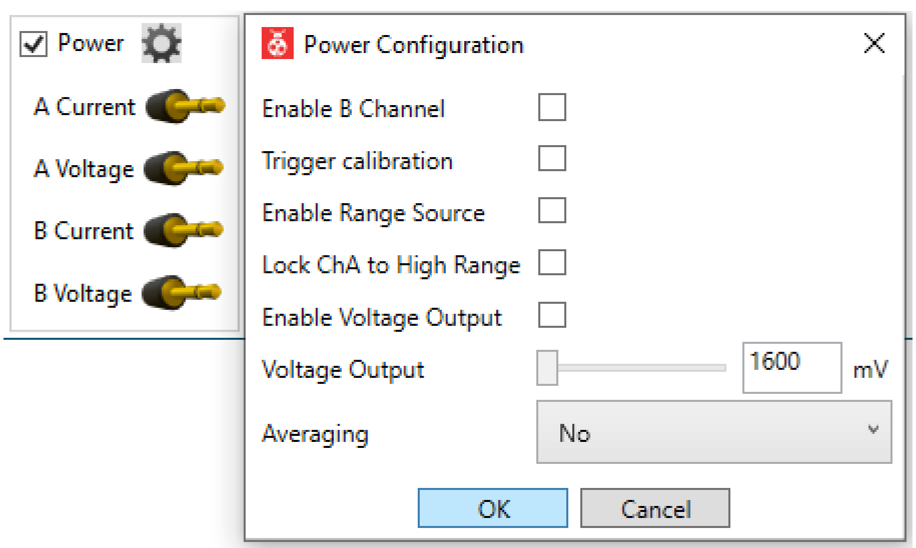
Press **Connect** and several interfaces will appear.

Figure 3-3. Power Debugger Interfaces



Enable the Power interface by pressing the empty box on the top left and then open the Settings menu.

Figure 3-4. Power Configuration

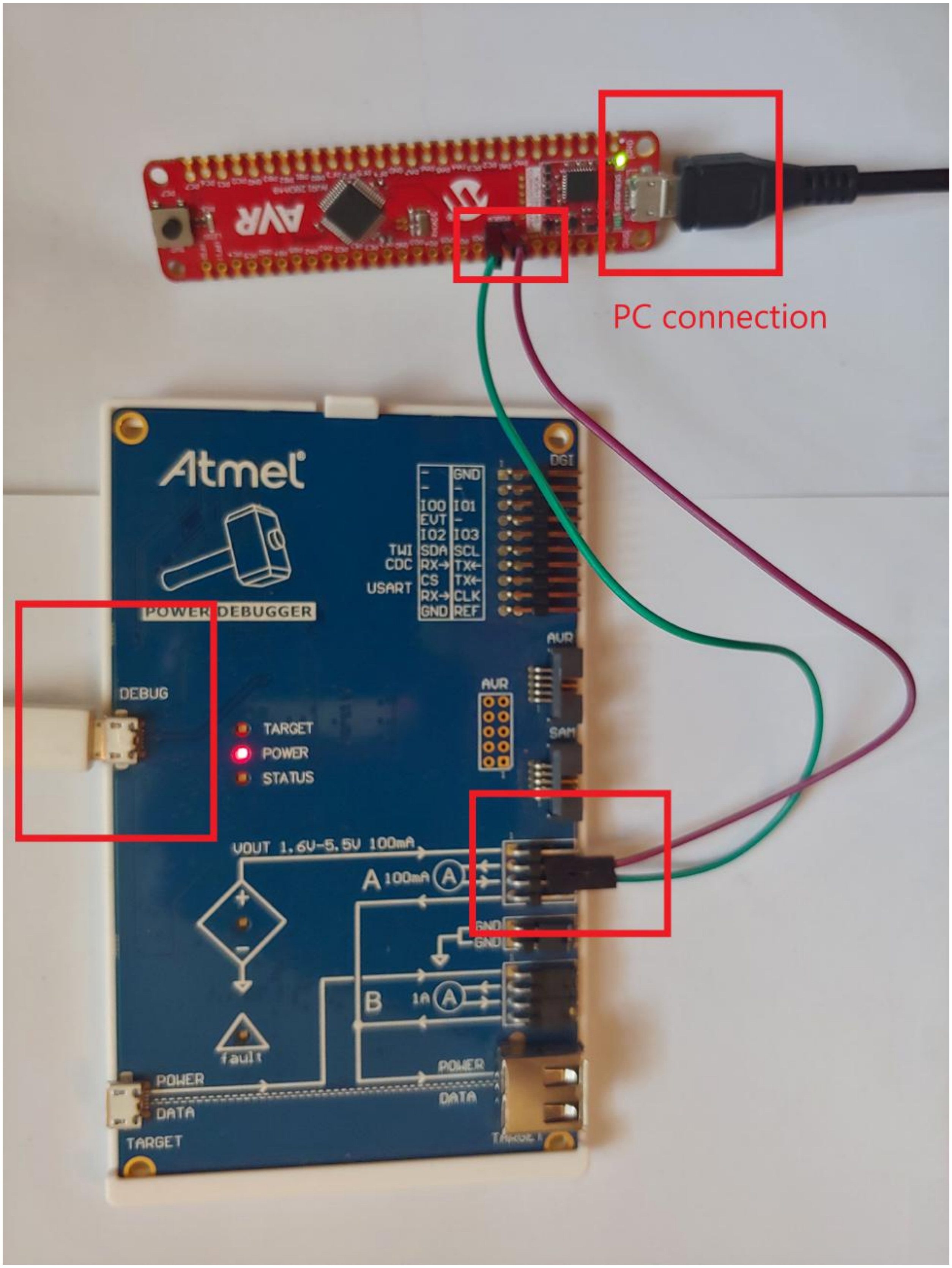


In the **Settings** tab, disable the B channel. The devices can also be powered from the Power Debugger by enabling the output voltage and setting it at the required value.

For this application, only the ammeter function of the Power Debugger is needed. Connect the line going into the A ammeter with the pin closest to the Nano Embedded Debugger on the Curiosity Nano board and the line going from the ammeter to the pin closest to the microcontroller on the Curiosity Nano.

When everything is correctly configured, press **Start** in Atmel Studio DGI control panel and watch the current consumed in real-time by the device.

Figure 3-5. Power Debugger Connection



4. Testing the Power Modes of the Microcontroller

The power consumed in the various modes is shown here with the use of a program controlled through USART. This program prints a menu at start-up, and the user can select one of the available functions by pressing the corresponding key.

The user can choose between six functions. The first four functions set a mode to test static consumption. The last two functions are more complex. One of them cycles through the available modes and stays for four seconds in each of them. The other one simulates the long deep sleep and short wake-up operation of battery-powered sensor applications.

The application is configured with MCC in MPLAB® X and can be found in this GitHub repository.

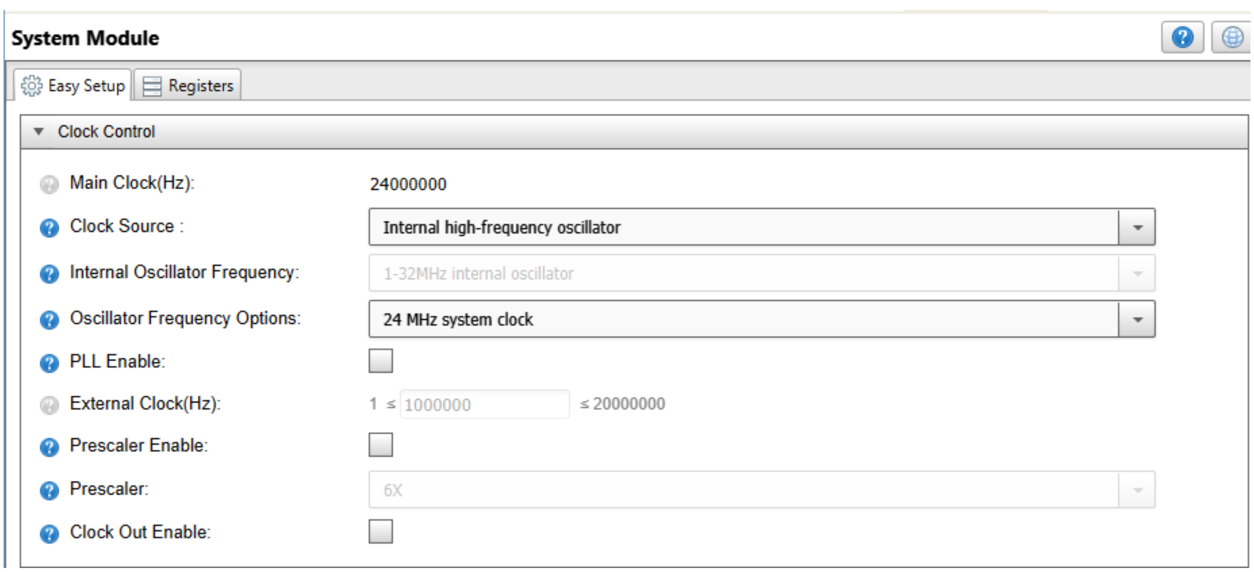
View Code Used for Testing on GitHub
Click to browse repositories

4.1 MCC Configuration

System Module

- Clock source: Internal high-frequency oscillator
- Oscillator frequency: 24 MHz
- Prescaler: Disabled

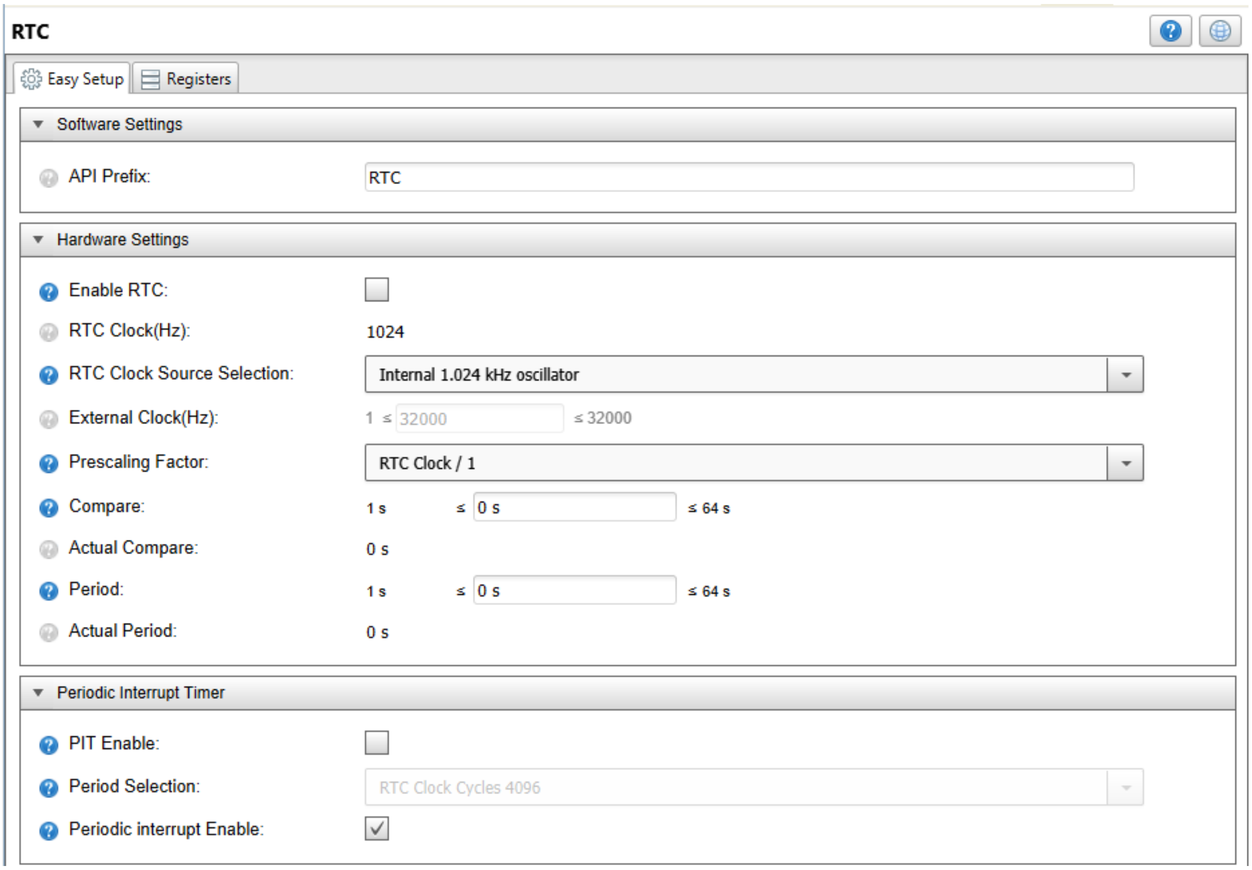
Figure 4-1. System Module Configuration



RTC

- RTC Clock Source Selection: Internal 1.024 kHz oscillator
- PIT: Periodic Interrupt Enable: Checked

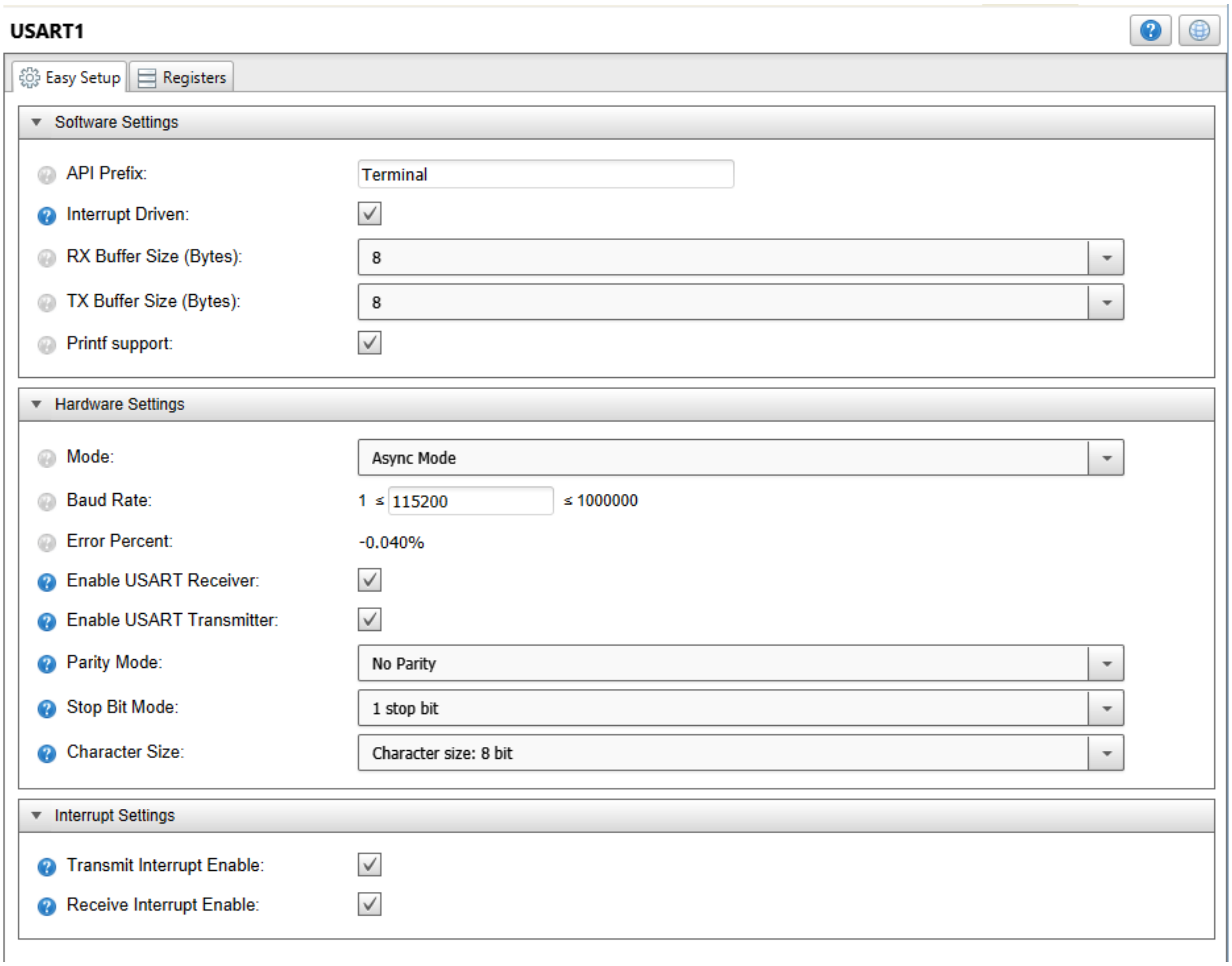
Figure 4-2. RTC and PIT Configuration



USART

- API Prefix: Terminal
- Interrupt Driven: Checked
- Printf Support: Checked
- Baud Rate: 115200
- Transmit Interrupt Enable: Checked
- Receive Interrupt Enable: Checked

Figure 4-3. USART Configuration



Pin Module and Manager

- PC0 as output TXD
- PC1 as input RXD
- PC6 set as output and Start High
- PC7 named 'Button', Pull-up enabled and Interrupt-on-Sense both edges

Figure 4-4. Pin Module Configuration

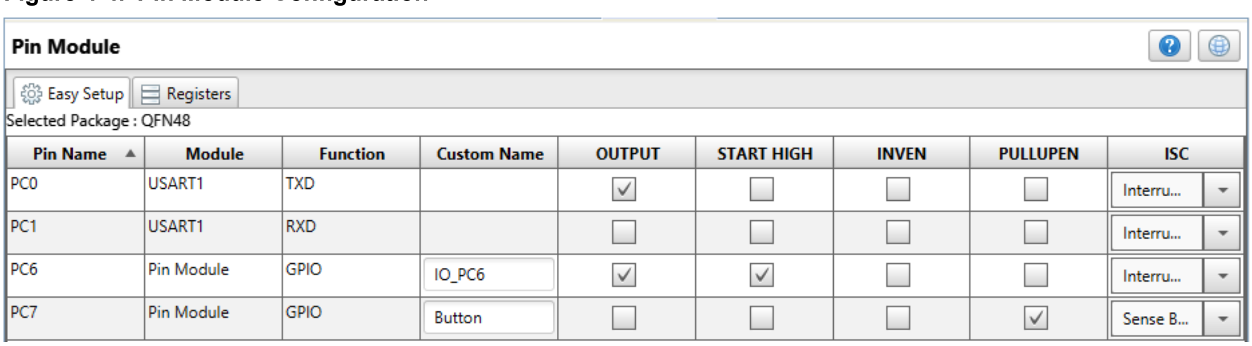
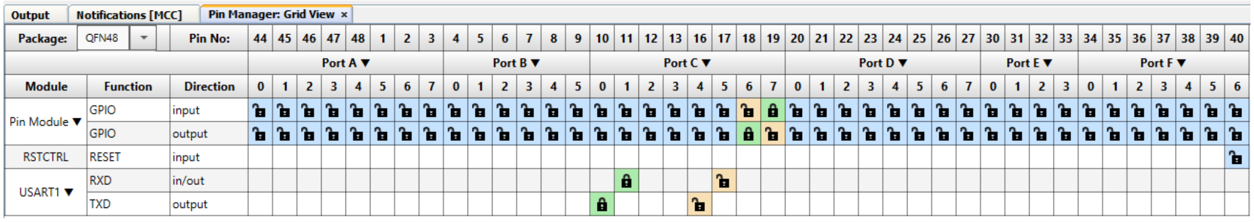


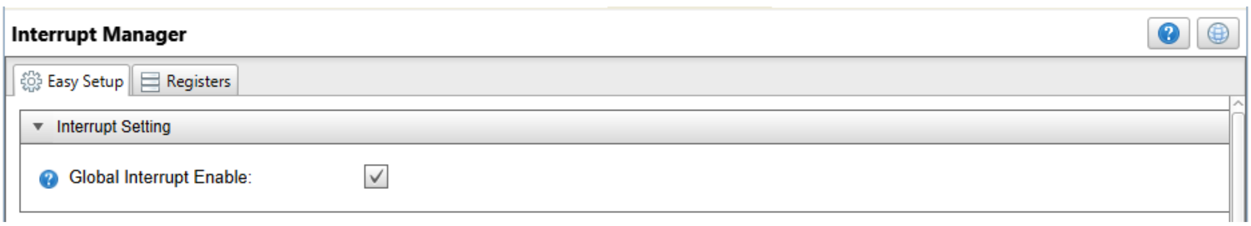
Figure 4-5. Pin Manager: Grid View



Interrupt Manager

- Global Interrupt Enable: Checked

Figure 4-6. Interrupt Manager Configuration



In the generated `pin_manager.c` file, the following rows are modified to set all the unused pins to output so that the power readings are not perturbed.

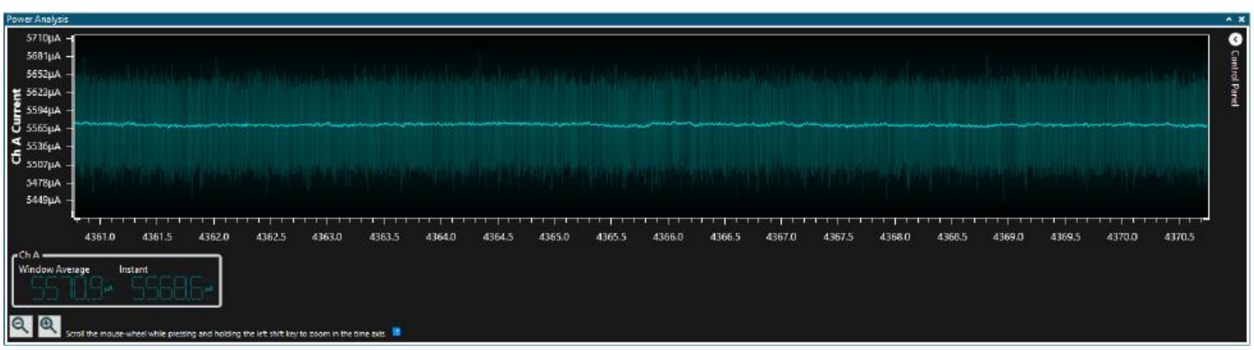
```
/* DIR Registers Initialization */
PORTA.DIR = 0xFF;
PORTB.DIR = 0xFF;
PORTC.DIR = 0x7D;
PORTD.DIR = 0xFF;
PORTE.DIR = 0xFF;
PORTF.DIR = 0xFF;
```

The `main.c` file contains the code for the various instructions that can be called from USART.

Active Mode

This mode is accessed by sending an 'a' or an 'A' character through USART. It keeps the microcontroller active and can be left by sending a new character.

Figure 4-7. Power Consumed in Active Mode

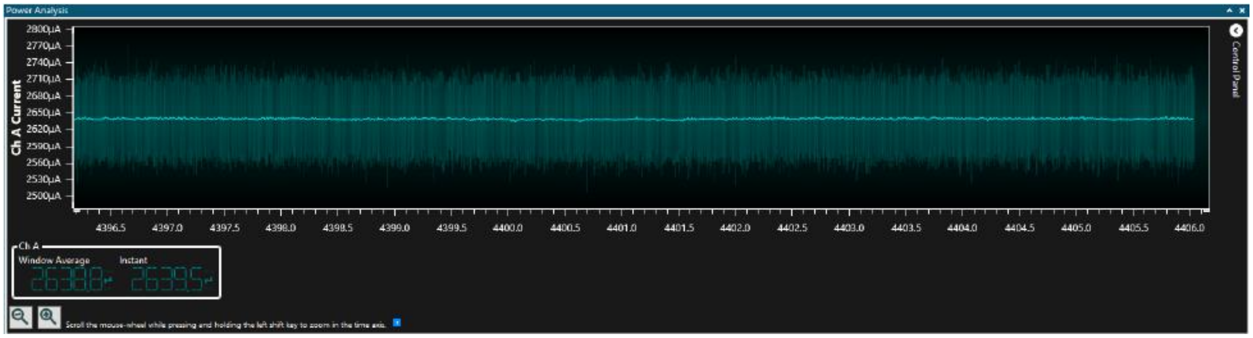


This is the average power consumed when the high-frequency oscillator is running at full power.

Idle Sleep Mode

This mode is accessed by sending an 'i' or 'I' character through USART. It puts the microcontroller into Idle sleep mode where the high-frequency oscillator is set to a lower consumption mode, but it is still operational and ready to act when the microcontroller exits a sleep mode. This mode can be left by sending a new character through USART.

Figure 4-8. Power Consumed in Idle Sleep Mode

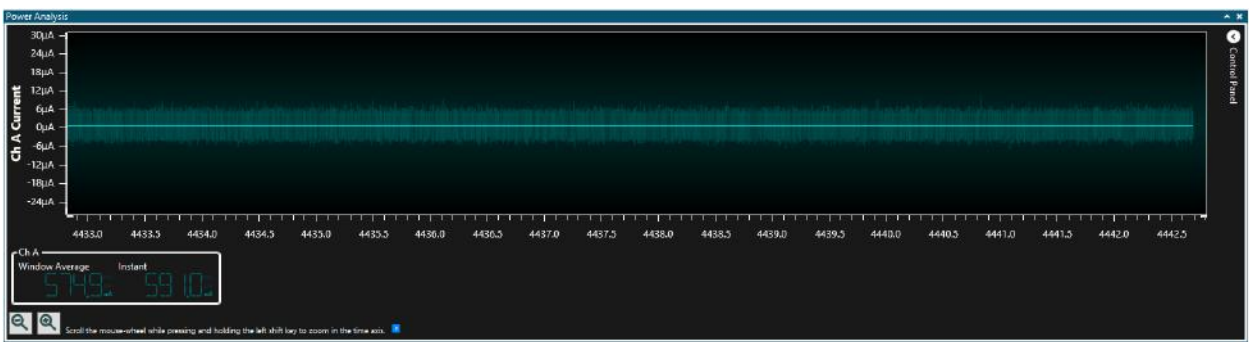


The power consumed by the high-frequency oscillator in Idle sleep mode is approximately half the power consumed in Active mode. If more peripherals are running in this mode, then the power consumption will be higher.

Standby Sleep Mode

This mode is accessed by sending an 's' or 'S' character through USART. It puts the microcontroller into Standby sleep mode. No peripherals are running in Standby sleep mode in this program, so the consumed power is just due to the oscillators. To exit this mode, the on-board button must be pressed, and then, a new character must be sent.

Figure 4-9. Power Consumed in Standby Sleep Mode



The power consumed in this mode is highly dependent on the number of peripherals that are running. In this example, no peripherals are running, so the consumed power is similar to the Power-Down sleep mode.

Power-Down Sleep Mode

This mode is accessed by sending a 'p' or 'P' character through USART. It puts the microcontroller into Power-Down sleep mode. The high-frequency oscillator is turned off, and only the 32.768 kHz internal oscillator is running. To exit this mode, the on-board button must be pressed, and then, a new character must be sent.

Figure 4-10. Power Consumed in Power-Down Sleep Mode

The power consumed in this mode is approximately the same as the power consumed in Standby sleep mode without active peripherals.

Cycle Instruction

This instruction is started by sending a 'c' or 'C' character through USART. It will cycle through all the previous modes with a four-second interval between them timed by the PIT. This instruction runs to completion and cannot be stopped until then. After it ends, a new character can be sent.

Figure 4-11. Power Consumed in Each Mode Relative to Each Other

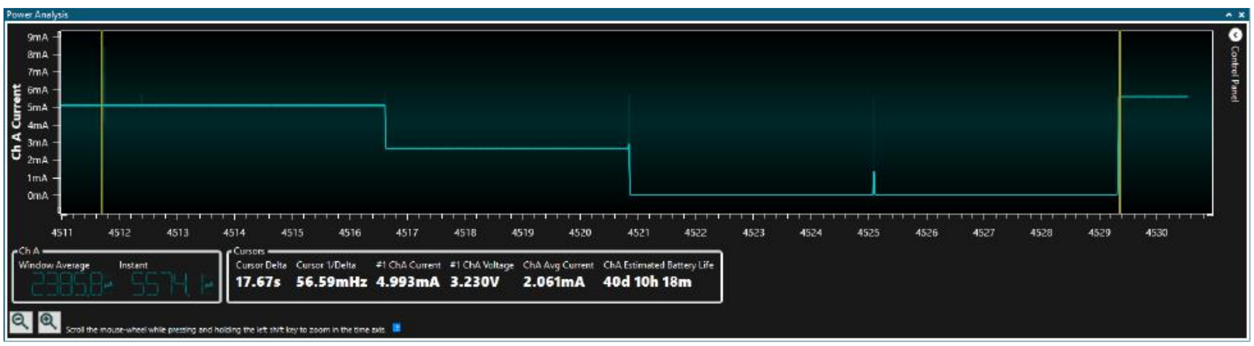
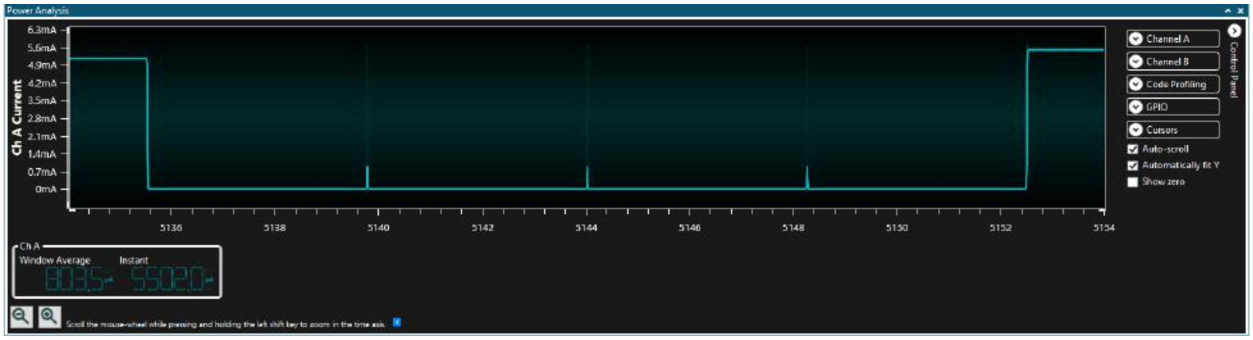


Figure 4-11 shows the differences between the power consumption of various modes. Idle sleep mode consumes about half the power the Active mode does, while Standby and Power-Down sleep modes are almost nothing compared to the other ones.

Deep Sleep and Short Wake-up Instruction

This instruction starts by sending a 'w' or 'W' character through USART. It will simulate a typical sensor application powered by a battery. It spends four seconds in Power-Down sleep mode and then sends a message that simulates reading sensors for a short time. This cycle repeats four times and is timed by the PIT. This instruction runs to completion and cannot be stopped until then. After it ends, a new character can be sent.

Figure 4-12. Power Consumed when Waking up to Read Sensors



The short current spikes represent the times the microcontroller wakes up to read the sensors.

Default Command

Any other character except the ones presented before will make the microcontroller print the menu with the available commands again.

5. Tips and Tricks for Reducing Power Consumption

The best way to reduce power consumption is to put the microcontroller to sleep whenever code does not need to be run. Identify the downtime of the code and put the microcontroller to sleep whenever possible.

Code execution implies that the microcontroller is awake, so the more that can be minimized, the more power will be saved. Several peripherals can become helpful for this as well as some hardware features for other peripherals.

1. **The Event System**
   One of the most helpful peripherals for power saving is the Event System. This peripheral offers a way to connect signals between peripherals without using code.

   Peripherals can generate events that are similar to interrupt conditions but do not wake up the device. These events can be connected through the Event System to activate the functions of other peripherals.

   For example, a timer overflow will produce an event that can be routed to the ADC, so a conversion starts whenever the condition is reached.

   More information on the Event System can be found in the data sheet for the AVR DA.
2. **Hardware Features**
   Many peripherals have hardware implementation for functions that would normally be done in software.

   Here are some examples of the most useful functions that can be done from the hardware.
   – The ADC has a hardware accumulation function that can go up to 128 samples. The ADC can also be put into Free-Running mode, where it starts another conversion whenever one ends.
   – Two timer B instances can be cascaded to obtain a 32-bit timer
   – USART offers error detection and error correction capabilities
   – TWI offers address match capabilities in any sleep mode

   Additionally, the Analog Comparator has power-saving modes that can reduce the power consumption at the expense of switching speed. This feature is found in the Control A (CTRLA) register of the AC.

   Figure 5-1. Power Profiles for Analog Comparator

   Bits 4:3 – POWER[1:0] Power Profile
   This setting controls the current through the comparator, which allows the AC to trade power consumption for the response time. Refer to the *Electrical Characteristics* section for power consumption and response time.

Value	Name	Description
0x0	PROFILE0	Power profile 0. Shortest response time and highest consumption.
0x1	PROFILE1	Power profile 1
0x2	PROFILE2	Power profile 2
0x3	-	Reserved

3. **The Configurable Custom Logic (CCL) Module**
   This module can be used to produce custom logic functions directly in hardware. Tasks that are usually done in software can be assigned to this peripheral, and the microcontroller core can be put to sleep.

   For example, an application that sends an SOS signal can be implemented in software with delays and pin changes, or a custom circuit that produces the SOS signal can be created with the CCL. More information can be found in the *Core Independent Solution Using AVR DA Peripherals Lab Training Manual*.

6. Conclusion

When designing applications where it is necessary to have the lowest possible power consumption, there are several design considerations to take into account.

The microcontroller must spend as little time as possible with the CPU running. The application must stay in a sleep mode as much as possible, wake up quickly to execute a task, and then go back to sleep again.

The three sleep modes, Idle, Standby, and Power-Down, in the AVR DA microcontrollers make it very flexible in designing low-power applications. Also, the Event System enabling inter-peripheral communication makes it possible for the peripherals to trigger actions between each other without involving the CPU. That way, the CPU can continue to stay in a sleep mode using zero power or executing other tasks.

Low-power designs are enabled by taking advantage of the features of the sleep controller as well as the inter-peripheral communication while the CPU is sleeping.

7. References

1. AVR128DA28/32/48/64 Data Sheet
2. Getting Started with CCL
3. AVR® DA Training Manuals:
   - Low-Power Modes Using Curiosity Nano
   - Differential ADC Using the AVR128DA48 Curiosity Nano
   - Core Independent Solution Using AVR DA Peripherals Lab
4. Code examples for AVR® DA

8. Revision History

Doc. Rev.	Date	Comments
A	10/2020	Initial document release

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **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 (FAQs), technical support requests, online discussion groups, Microchip design partner 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

Product Change Notification Service

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

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: 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 specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "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: 978-1-5224-6812-7

Quality Management System

For information regarding Microchip’s Quality Management Systems, please visit www.microchip.com/quality.

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: 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: 45-4485-5910 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: 49-7131-72400 **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: 47-72884388 **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