TB3234

Internal High-Frequency Oscillator Calibration Using the Auto-Tune Feature

Introduction

Author: Ioan Pop, Microchip Technology Inc.

The purpose of this document is to provide more details on the internal high-frequency oscillator (OSCHF) auto-tune feature of the AVR® DA MCU (AVR DA) family and how to activate it. The auto-tune feature allows OSCHF to adjust its frequency, comparing it to an external 32.768 kHz crystal oscillator to better match the desired value.

This document describes four use cases in which the internal high-frequency oscillator will be used to drive the main clock, each use case containing two scenarios. For the first three use cases the two scenarios are: with auto-tune feature activated and deactivated. The purpose of the last use case is to check the auto-tune feature against an incorrect tuning value provided by the user, therefore the two scenarios for the last use case are: incorrect tune value setting and auto-tune enabled. The switch between the two scenarios is made using the PC7 pin (the on-board SW0 button of the AVR128DA48 Curiosity Nano board). Additionally, the main clock will be output to the CLKOUT pin (PA7) in order to showcase the auto-tune feature improvement on the OSCHF output frequency precision. The use cases described in this technical brief are:

- **Configure OSCHF to run at 1 MHz and activate/deactivate the auto-tune feature:**
  The purpose of this use case is to configure the OSCHF to run at 1 MHz and drive the main clock, and enable frequency output on the CLKOUT pin. Two frequency measurements on the CLKOUT pin will be made using an oscilloscope - with auto-tune feature activated and deactivated.
- **Configure OSCHF to run at 4 MHz and activate/deactivate the auto-tune feature:**
  The purpose of this use case is to configure the OSCHF to run at 4 MHz and drive the main clock, and enable frequency output on the CLKOUT pin. Two frequency measurements on the CLKOUT pin will be made using an oscilloscope - with auto-tune feature activated and deactivated.
- **Configure OSCHF to run at 24 MHz and activate/deactivate the auto-tune feature:**
  The purpose of this use case is to configure the OSCHF to run at 24 MHz and drive the main clock, and enable frequency output on the CLKOUT pin. Two frequency measurements on the CLKOUT pin will be made using an oscilloscope - with auto-tune feature activated and deactivated.
- **Configure OSCHF to run at 4 MHz with incorrect tuning value:**
  The purpose of this use case is to configure the OSCHF to run at 4 MHz and drive the main clock, and enable frequency output on the CLKOUT pin. An error injection will be made to the tune register to check if the auto-tune mechanism will be able to correct it. Two frequency measurements on the CLKOUT pin will be made using an oscilloscope - with the incorrect frequency tune input and with auto-tune feature activated

Note: The AVR128DA48 Curiosity Nano was used for the tests performed and described in this document.

Table of Contents

1. Relevant Devices

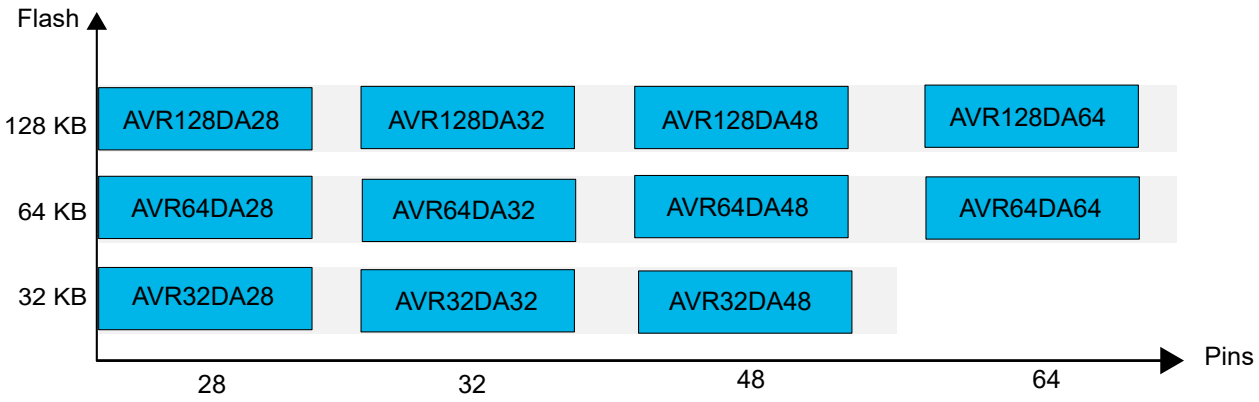
This chapter lists the relevant devices for this document.

1.1 AVR® DA Family Overview

The figure below shows the AVR® DA devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore, the available features.

Figure 1-1. AVR® DA Family Overview



Devices with different Flash memory size typically also have different SRAM.

2. Overview

The auto-tune feature is enabled by setting the AUTOTUNE bit in the internal high-frequency oscillator Control A (CLKCTRL.OSCHFCTRLA) register. For as long as the bit is set, the auto-tune feature compares the internal high-frequency oscillator to a 1.024 kHz reference from the external crystal oscillator. If the calibration is performed, the auto-tune system checks for a clock drift every 64 ms. The value required for the tuning is stored in the internal high-frequency oscillator Frequency Tune (CLKCTRL.OSCHFTUNE) register when the auto-tune feature is turned off.

The tuning register has 6 bits that provide 64 steps for adjusting the frequency. The bits for the up/down adjustment are stored in two's complement. The OSCHFTUNE register has 8 bits where the first 5 bits represent the value, the sixth one is the sign, and the seventh and the eight are a mirror to the sign. The tuning register starts with the default value of `0x00`. One step increases or decreases the clock speed by a percentage of the clock. If the error between the reference and the clock is less than one step, then the auto-tune feature will not activate, yet if the difference is more significant, it will activate and make the approximation for the next step. For example, if the default value of the clock is close to the correct value, auto-tune will not modify the default value, but if the clock is tuned into a different incorrect value, the auto-tune feature might set the tuning register to one step from the default value, for example, `0x01` instead of `0x00`.

3. Activating and Testing the Auto-Tune Feature

In order to use the auto-tune feature, a high precision 32.768 kHz external crystal oscillator is required.

The bits that need to be set in order to enable this feature are under Configuration Change Protection. These are I/O registers, so the I/O signature, `0xD8`, needs to be written in the Configuration Change Protection (CPU.CCP) register.

In order to write to a register under I/O Configuration Change Protection, the _PROTECTED_WRITE macro must be used, which will write the signature to the CPU.CCP register and then the value to the desired register:

```
_PROTECTED_WRITE (register, value);
```

Replace ‘register’ with the register that needs to be written and ‘value’ with the value that needs to be written to it.

Note: The compiler optimization level must be set to at least O1 in the Atmel Studio project settings. Oterwise, the macro will not be able to write the value.

This crystal oscillator needs to be enabled, by writing the ENABLE bit from the 32.768 kHz crystal oscillator Control A (CLKCTRL.XOSC32KCTRLA) register to ‘1’. The following line of code enables the oscillator:

```
_PROTECTED_WRITE (CLKCTRL.XOSC32KCTRLA, CLKCTRL_ENABLE_bm);
```

After this is done, the auto-tune feature needs to be enabled by setting the AUTOTUNE bit (bit 0) in the internal high-frequency oscillator Control A (CLKCTRL.OSCHFCTRLA) register. This register has two other features. The RUNSTDBY bit (bit 7) activates the running in Standby mode and the FRQSEL bitfield (bit 2 to bit 5) sets the frequency of the oscillator. The next line of code enables the auto-tune feature and sets the frequency to 1 MHz. The frequency can be changed by giving different values that will be detailed later in this document:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, (CLKCTRL_FREQSEL_1M_gc | CLKCTRL_AUTOTUNE_bm));
```

The internal high-frequency oscillator must be selected as the main clock. This is done by writing the value that selects the internal high-frequency oscillator to the CLKSEL bit field in the Main Clock Control A (CLKCTRL.MCLKCTRLA) register. Setting the CLKOUT bit in this register outputs the clock signal on the PA7 pin. The next two lines of code show how to select the internal high-frequency oscillator and how to enable the clock output signal:

```
_PROTECTED_WRITE (CLKCTRL.MCLKCTRLA, CLKCTRL_CLKSEL_OSCHF_gc);
```

```
_PROTECTED_WRITE (CLKCTRL.MCLKCTRLA, (CLKCTRL_CLKSEL_OSCHF_gc | CLKCTRL_CLKOUT_bm));
```

Note: The default startup oscillator is configured using FUSE.OSCCFG fuse. If the internal high-frequency oscillator is already set as default, then this step may be omitted. For more details on Fuse settings, see ‘FUSE - Configuration and User Fuses’ chapter of the AVR128DA48 Data Sheet.

If the only requirement is the turning on of the auto-tune feature, the following code provides this functionality:

```
_PROTECTED_WRITE (CLKCTRL.XOSC32KCTRLA, CLKCTRL_ENABLE_bm);
_PROTECTED_WRITE (CLKCTRL.MCLKCTRLA, CLKCTRL_CLKSEL_OSCHF_gc);
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, (CLKCTRL_FREQSEL_1M_gc | CLKCTRL_AUTOTUNE_bm));
```

Note: The required frequency is also set through the OSCHFCTRLA register, so the value needs to be changed to the desired one. The register setting from above sets it to 1 MHz.

In order to highlight the auto-tune feature, several tests were performed at different frequencies and the results are provided in each use case chapter.

The image captures were done using a Tektronix MDO3024 Mixed Domain Oscilloscope with the following settings:

- Gating set to Screen
- High-Low Method set to Histogram
- Coupling in DC
- Bandwidth Full

The testing method involves setting up the register for the correct frequency and using a button that activates the auto-tune feature, waits for one second, and then stops it. The tuning (CLKCTRL.OSCHFTUNE) register can then be read in Debugging mode to check the value put there by the auto-tune feature.

The mean and standard deviation values were calculated using 100 samples. The mean is the value that best indicates the clock frequency, because at high frequencies, random noise from the environment makes the instant reading inaccurate.

4. Configure OSCHF to Run at 1 MHz and Activate/Deactivate the Auto-Tune Feature

The following code example starts the internal high-frequency oscillator at the desired frequency and sets up an interrupt on a rising edge of the signal coming from the button. When the interrupt is triggered, the auto-tune feature is enabled, the microcontroller waits for a second and then deactivates it in order to check the value the auto-tune feature stores in the tuning register.

```
#define F_CPU                                   1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define DELAY_TIME                              1000
#define PRESSED                                 1
#define NOT_PRESSED                             0
#define PULL_UP_ENABLE                          0x08
#define BUTTON_PIN                              PIN7_bm

void CLK_init(void);
void PORT_init(void);

uint8_t volatile button_event = NOT_PRESSED;

int main(void)
{
    cli();
    CLK_init();
    PORT_init();
    sei();

    while (1)
    {
        if(button_event == PRESSED)
        {
            cli();
            _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA,(CLKCTRL_FREQSEL_1M_gc|CLKCTRL_AUTOTUNE_bm));
            _delay_ms(DELAY_TIME);
            _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_1M_gc);
            button_event = NOT_PRESSED;
            sei();
        }
    }
}

void CLK_init(void)
{
    _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_1M_gc);
    _PROTECTED_WRITE (CLKCTRL.XOSC32KCTRLA, CLKCTRL_ENABLE_bm);
    _PROTECTED_WRITE (CLKCTRL.MCLKCTRLA, (CLKCTRL_CLKSEL_OSCHF_gc | CLKCTRL_CLKOUT_bm));
}

void PORT_init(void)
{
    PORTC.DIRCLR = BUTTON_PIN;
    PORTC.INTFLAGS = BUTTON_PIN;
    PORTC.PIN7CTRL = PORT_ISC_RISING_gc | PULL_UP_ENABLE;
}

ISR(PORTC_PORT_vect)
{
    button_event = PRESSED;
    PORTC.INTFLAGS = BUTTON_PIN;
}
```

View Code Example on GitHub
Click to browse repository

Tip: The full code example is also available in 10. Appendix.

The PORT_init() function configures Port C Pin 7 (PC7) as an input, enables the internal pull-up and the PORTC interrupt for external GPIO events such as pressing a button.

The CLK_init() function configures all the registers that select the main clock for 1 MHz operation, enables the 32.768 kHz external crystal oscillator, and enables the output of the main clock on CLKOUT pin (PA7).

The Interrupt Service Routine (ISR) for the button sets a flag that is polled in the main loop.

The sei() command enables global interrupts and cli() disables them. This is done to prevent unwanted interrupts from disrupting the _PROTECTED_WRITE macro.

The main loop enables the auto-tune feature for one second, then disables it. This ensures there is enough time for it to correct the error and that the value is stored in the tuning register at the end. Depending on the noise present at the interface between the 32.768 kHz external crystal oscillator and the microcontroller, the tuning time can be higher than one second, but in practice, this time is sufficient under most normal conditions. If the signal is too distorted, as in the case when a finger is placed on the traces or the pins of the microcontroller, the auto-tune feature will not be able to run properly.

The following pictures were obtained using the oscilloscope; Figure 4-1 with normal start-up frequency and Figure 4-2, after the auto-tune feature is enabled.

The line of the code for the correct frequency is:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_1M_gc);
```

The line for starting the auto-tune feature is:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, ((CLKCTRL_FREQSEL_1M_gc)|(CLKCTRL_AUTOTUNE_bm)));
```

Figure 4-1. 1 MHz Clock Without Auto-Tune

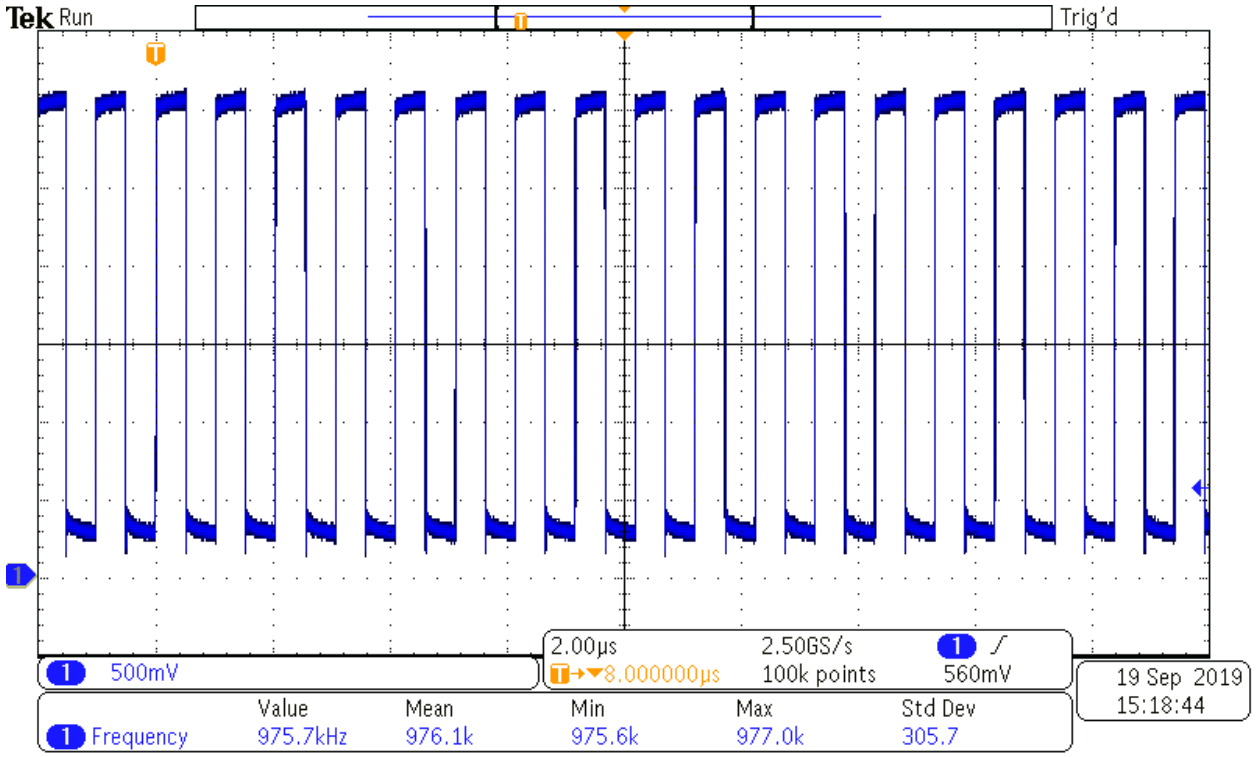
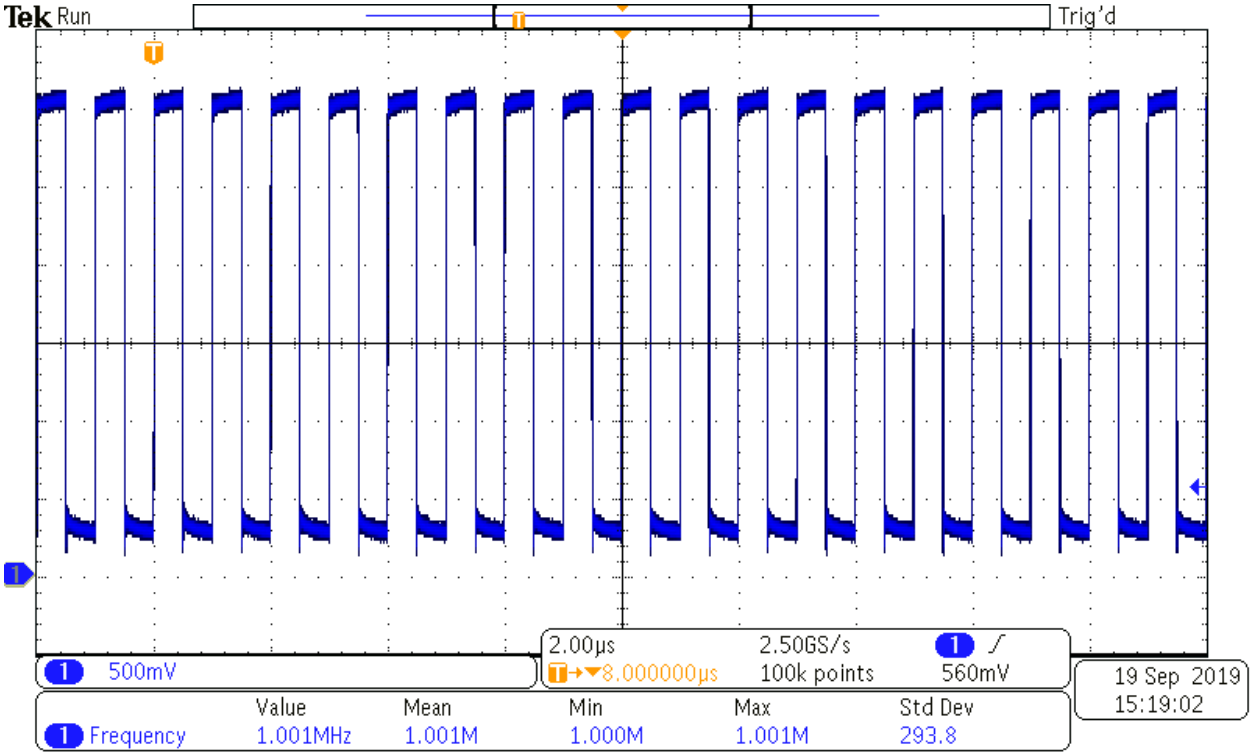


Figure 4-2. 1 MHz Clock With Auto-Tune



The value in the OSCHFTUNE register at the end of the auto-tune process is `0x06`.

As the figures show, the error has been corrected and the value is much closer to the required 1 MHz than it was before.

5. Configure OSCHF to Run at 4 MHz and Activate/Deactivate the Auto-Tune Feature

The default frequency value setting for the internal high-frequency oscillator is 4 MHz.

The following line of code configures the OSCHF to run at 4 MHz:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_4M_gc);
```

The following line of code enables the auto-tune feature:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, ((CLKCTRL_FREQSEL_4M_gc)|(CLKCTRL_AUTOTUNE_bm)));
```

Figure 5-1. 4 MHz Clock Without Tuning

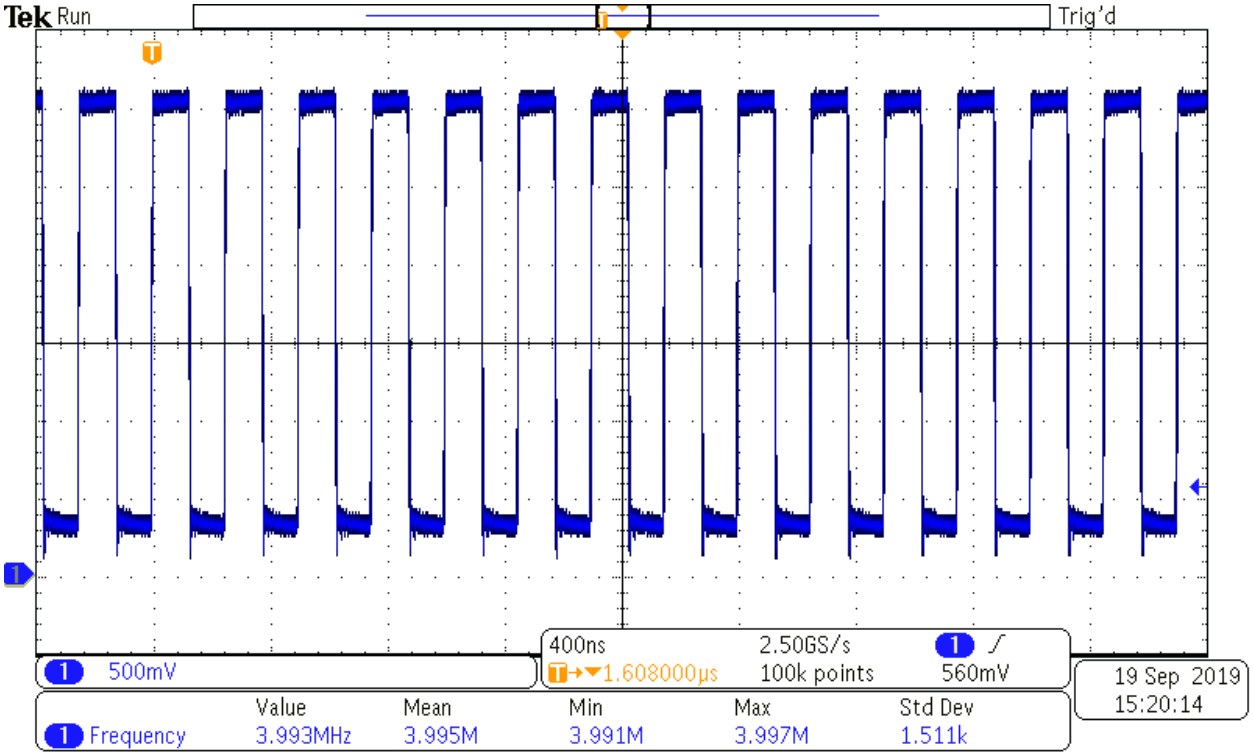
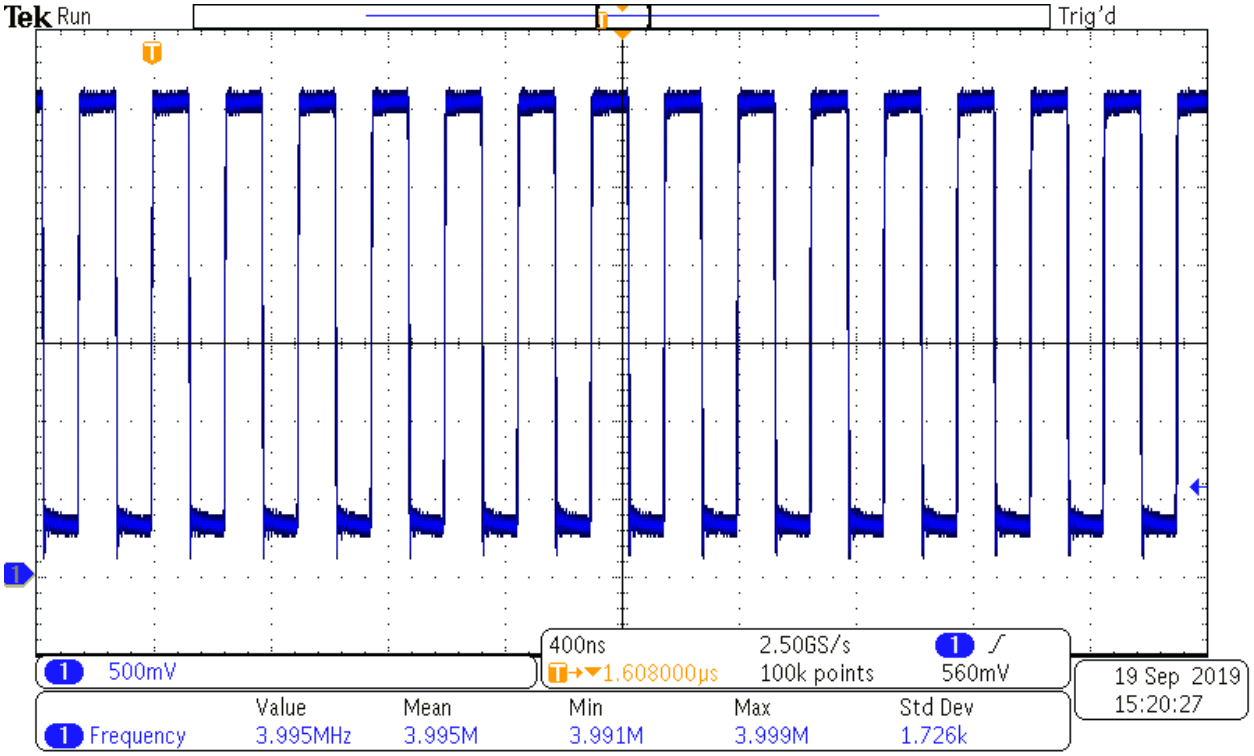


Figure 5-2. 4 MHz Clock After Auto-Tune



As can be observed from the figures, the error is too small for the auto-tune feature to correct. The value in the tuning register is `0x00`, the default at which it starts.

Note: The 4 MHz frequency is calibrated during fabrication. The higher values are derived from the 4 MHz one and are also calibrated.

6. Configure OSCHF to Run at 24 MHz and Activate/Deactivate the Auto-Tune Feature

The 24 MHz clock is the highest base frequency for the internal high-frequency oscillator.

The following line of code sets the clock speed of OSCHF to 24 MHz:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_24M_gc);
```

The following line of code for sets the OSCHF clock speed to 24 MHz and enables the auto-tune feature:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, ((CLKCTRL_FREQSEL_24M_gc)|(CLKCTRL_AUTOTUNE_bm)));
```

Figure 6-1. 24 MHz Clock Without Tuning

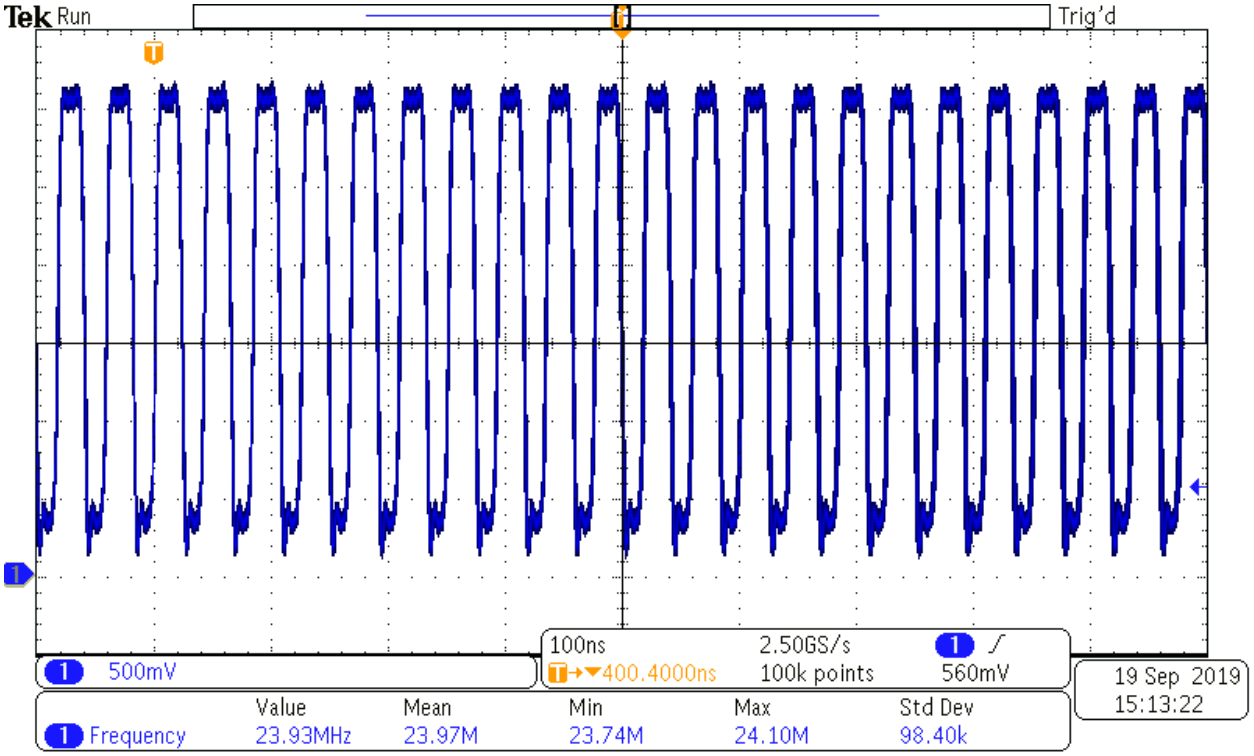
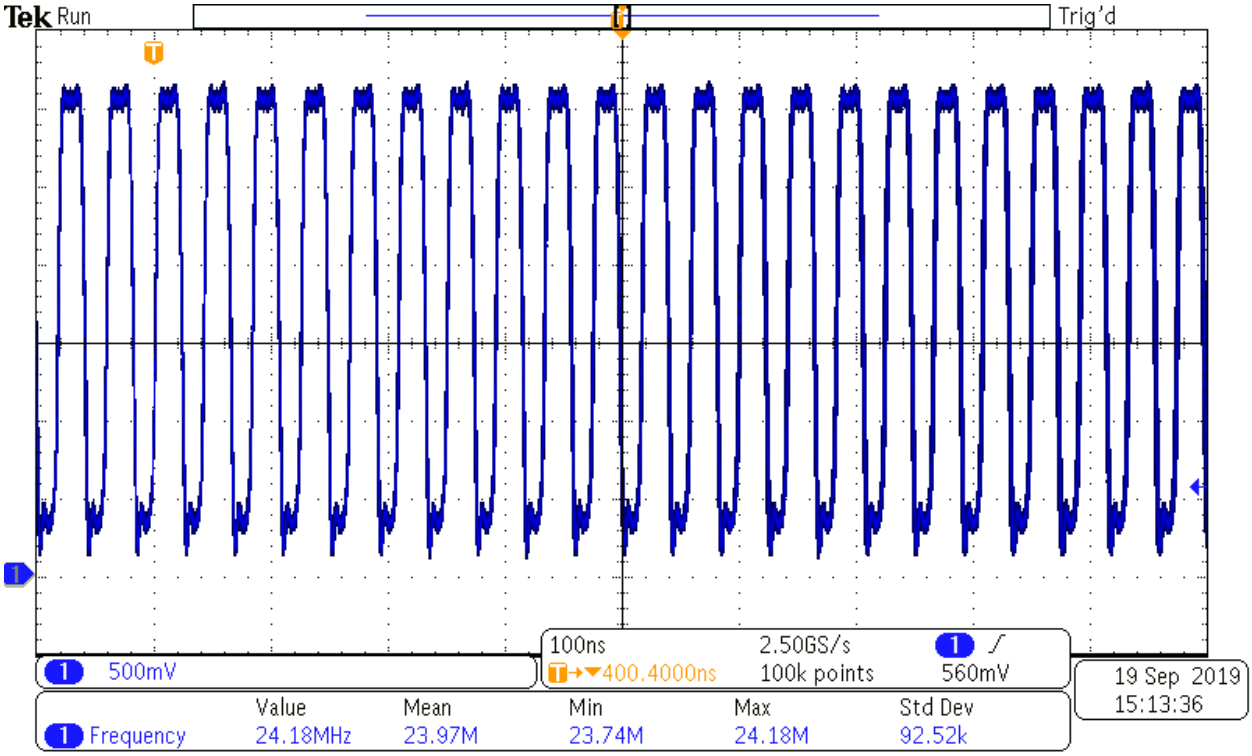


Figure 6-2. 24 MHz Clock After Auto-Tune



The figures above show the error was within the tolerated values to be corrected, thus the tuning register remained unchanged.

7. Configure OSCHF to Run at 4 MHz with Incorrect Tuning Value

To better highlight the auto-tune feature, an incorrect value, `0x0F` is placed in the OSCHFTUNE register. This increases the clock frequency to a level above the error threshold. The following line of code loads the OSCHFTUNE register with the desired value:

```
_PROTECTED_WRITE (CLKCTRL.OSCHFTUNE, 0x0F);
```

Figure 7-1 shows the CLKOUT when `0xF` is written into OSCHFTUNE register and Figure 7-2 displays the CLKOUT after the auto-tune feature compensated the frequency drift.

Figure 7-1. 4 MHz Clock With Incorrect Tuning Value

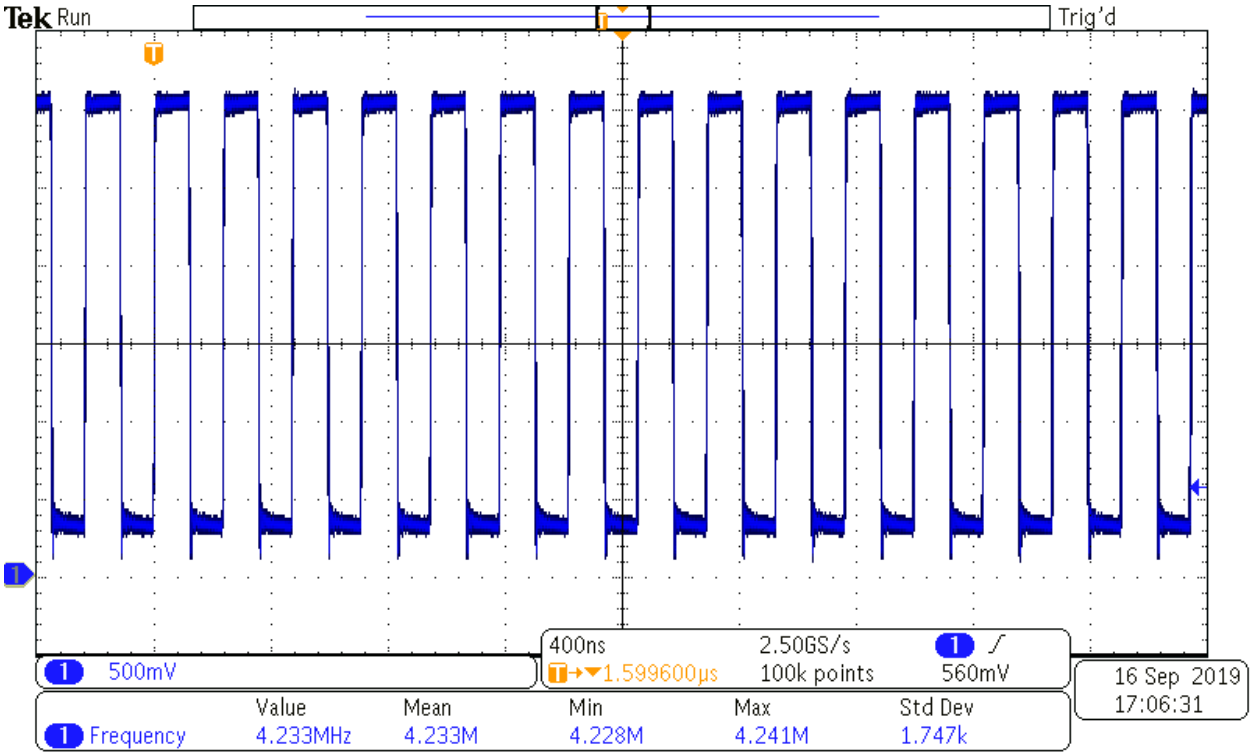
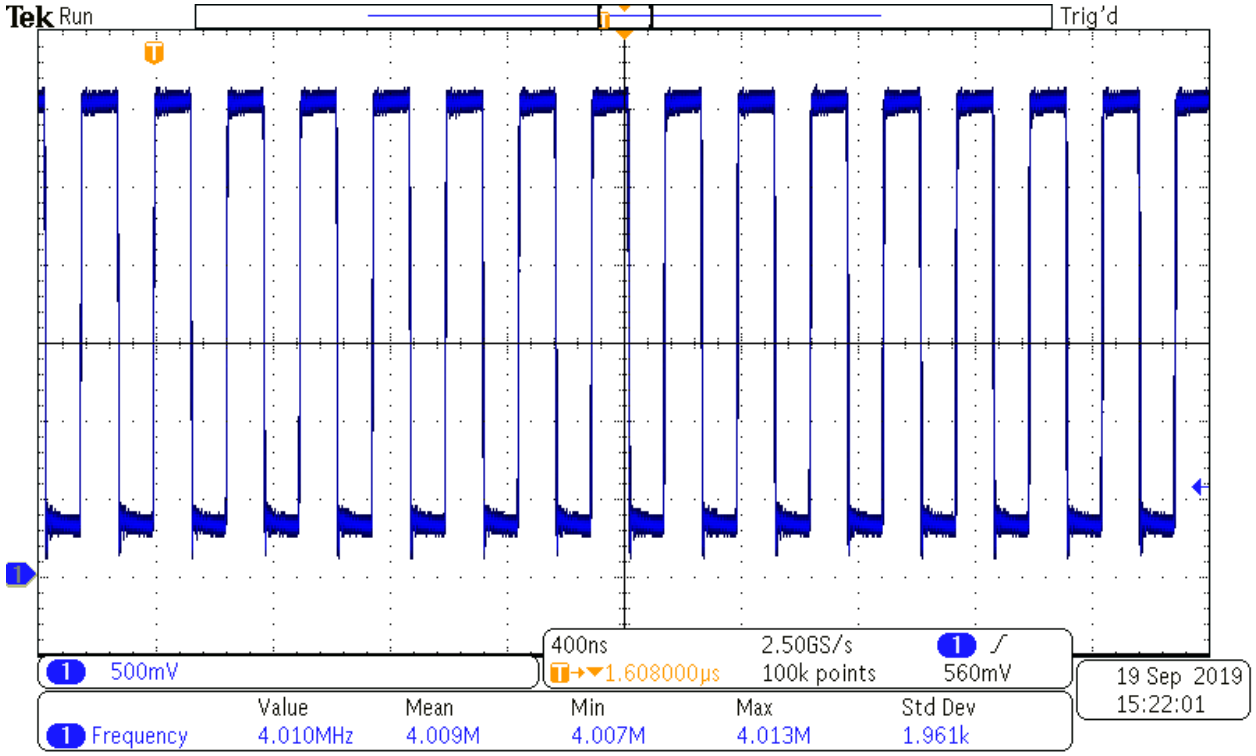


Figure 7-2. 4 MHz Clock After Auto-Tune from Incorrect Starting Value



At the end of the auto-tune procedure, the value in the tuning register is `0x01`. This is different from the previous test, where the value was `0x00`. As explained in the Overview section, this is normal behavior when the error is smaller than what a single step would correct.

8. Conclusion

In conclusion, the auto-tune feature is a simple, yet effective method to increase the internal high-frequency oscillator output frequency precision. It can be easily added to any project which relies on the OSCHF to provide a more accurate clock frequency to the CPU or peripherals.

9. References

1. *AVR128DA28/32/48/64 Preliminary Data Sheet*.
2. *AVR128DA48 Curiosity Nano User's Guide*.

10. Appendix

Example 10-1. Configure OSCHF to Run at 1 MHz and Activate/Deactivate the Auto-Tune Feature

```
#define F_CPU                                   1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define DELAY_TIME                              1000
#define PRESSED                                 1
#define NOT_PRESSED                             0
#define PULL_UP_ENABLE                          0x08
#define BUTTON_PIN                              PIN7_bm

void CLK_init(void);
void PORT_init(void);

uint8_t volatile button_event = NOT_PRESSED;

int main(void)
{
    cli();
    PORT_init();
    CLK_init();
    sei();

    while (1)
    {
        if(button_event == PRESSED)
        {
            cli();
            _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA,(CLKCTRL_FREQSEL_1M_gc|
CLKCTRL_AUTOTUNE_bm));
            _delay_ms(DELAY_TIME);
            _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_1M_gc);
            button_event = NOT_PRESSED;
            sei();
        }
    }
}

void CLK_init(void)
{
    _PROTECTED_WRITE (CLKCTRL.OSCHFCTRLA, CLKCTRL_FREQSEL_1M_gc);
    _PROTECTED_WRITE (CLKCTRL.XOSC32KCTRLA, CLKCTRL_ENABLE_bm);
    _PROTECTED_WRITE (CLKCTRL.MCLKCTRLA, (CLKCTRL_CLKSEL_OSCHF_gc |
CLKCTRL_CLKOUT_bm));
}

void PORT_init(void)
{
    PORTC.DIRCLR = BUTTON_PIN;
    PORTC.INTFLAGS = BUTTON_PIN;
    PORTC.PIN7CTRL = PORT_ISC_RISING_gc | PULL_UP_ENABLE;
}

ISR(PORTC_PORT_vect)
{
    button_event = PRESSED;
    PORTC.INTFLAGS = BUTTON_PIN;
}
```

11. Revision History

Doc. Rev.	Date	Comments
C	05/2020	Updated AVR® MCU DA (AVR-DA) to AVR® DA MCU and AVR-DA to AVR DA, per latest trademarking
B	03/2020	Updated repository links. Updated AVR-DA to AVR® MCU DA (AVR-DA), per latest trademarking.
A	02/2020	Initial document release

The Microchip Website

Microchip provides online support via our website at http://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 http://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: 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

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-6062-6

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://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: http://www.microchip.com/support Web Address: http://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