Replicating Simple Functions Using MSP430 Value Line MCUs

Mitch Ridgeway

MSP430 FRAM Applications

Abstract

Replicating Simple Functions Using MSP430 Value Line MCUs Summary:

TI has introduced a series of Tech Notes aimed towards realizing simple functions using the MSP430 Value Line Sensing family of microcontrollers (MCUs). By promoting programmability, small package sizes, ultra-low power characteristics, and aggressive retail prices, these functions showcase the MSP430's adaptability in a wide range of applications where embedded MCUs are not typically considered.

What you'll learn:

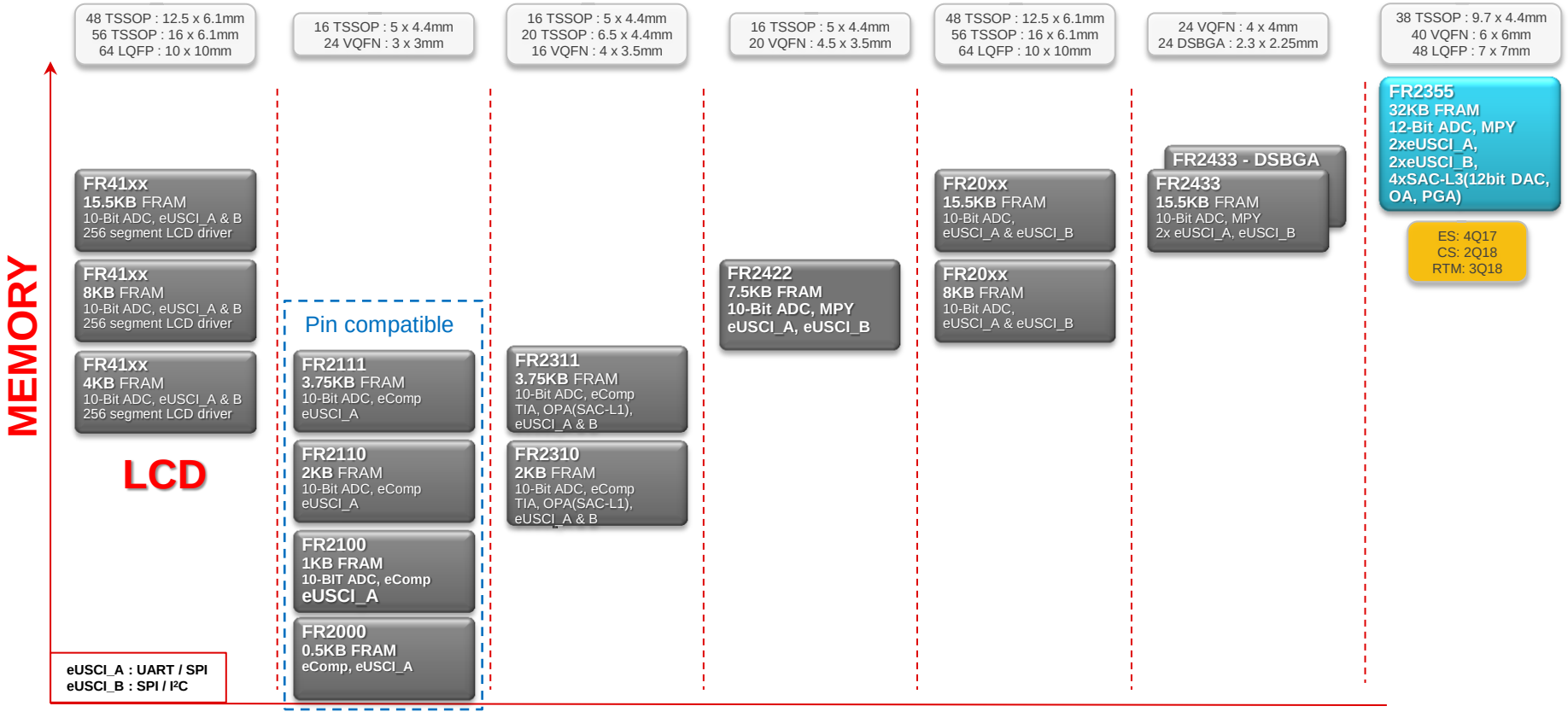
- Understand advantages of MCU adoption
- 25 Functions
- Navigate eBook
- Optimize code for the smallest memory footprint
- Use resources to develop additional functions

Detailed Agenda

- 25 Functions
  - Pulse Width Modulation (PWM)
  - Timer
  - System
  - Communications
- Code Optimization Tips & Tricks
  - IDE settings
  - Efficient coding techniques
  - Live example

Value Line Portfolio

Production | Sampling



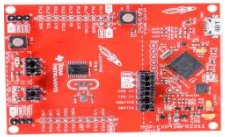
MSP430FR2000

Features/Benefits

- **FRAM: Ultra-low power, universal memory** – Nearly infinite (10^{15}) write cycles, >100x faster than Flash (2MB/s), 250x less power in writes, flexible as data or program memory

Tools

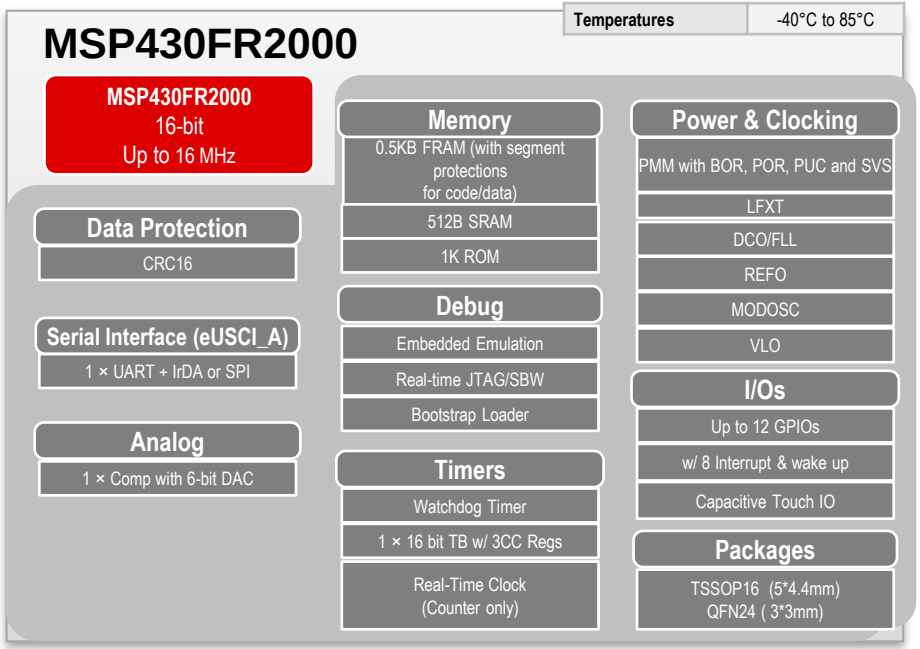
FR2311 LaunchPad



Target Board

Software

- **MSP430Ware**
- **Code Composer Studio™ IDE**
- **IAR**



Target Functions

- **UART Software Controlled RGB LED Color Mixing**
- **Voltage Monitor with Timestamp**
- **Single-Slope Analog-to-Digital Conversion Technique**
- **UART-to-SPI Bridge**

Why 25 Functions for 25 Cents?

- TI has introduced a series of Tech Notes aimed towards realizing simple functions using the MSP430 Value Line Sensing family of microcontrollers (MCUs). These applications are capable of addressing the conventional needs of end-equipments across the industrial, personal electronic, and telecommunication markets.
- Several fixed function ICs do not include the flexibility required for a user's application or might be offered at a price too high to be competitively considered in a system design
- Showcase the MSP430's ability to replace dedicated ICs in a wide range of applications where embedded MCUs are not typically considered
- Key Components
  - Programmability
  - Small package sizes
  - Ultra-low power
  - Aggressive retail prices

Function list for MCUs

PWM Functions

- UART Software Controlled RGB LED Color Mixing
- Stepper Motor Control
- Servo Motor Controller
- Dual-Output 8-Bit PWM DAC
- Analog Input to PWM Output

Timer Functions

- External RTC with Backup Memory
- Simple RTC-Based System Wake-Up Controller
- Programmable System Wake-Up Controller
- External Programmable Watchdog Timer
- 7-Segment LED Stopwatch

System Functions

- EEPROM Emulation
- Low Power Hex Keypad
- Single-Slope Analog-to-Digital Conversion Technique
- Multi-Function Reset Controller
- Hysteresis Comparator with UART
- ADC Wake and Transmit on Threshold
- Tamper Detection
- Quadrature Encoder Position Counter
- Programmable Clock Source
- Programmable Frequency Locked Loop

Communications

- Single-Wire Communication Host
- UART-to-UART Bridge with Baud Rate Translation
- UART-to-SPI Bridge
- SPI I/O Expander

Pulse Width Modulation (PWM)

UART Software Controlled RGB LED Color Mixing

Overview

This simple application uses a low cost MSP430FR2000 device to control Red Green Blue (RGB) LED color mixing. The mixing can be controlled externally by using a UART interface.

Resources

Hardware Peripherals
- Timer_B
- eUSCI_A (UART mode)

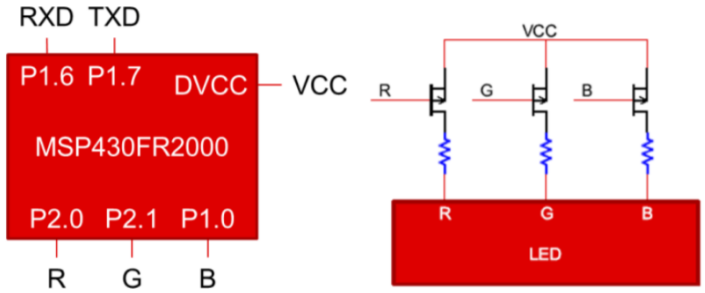
Firmware Footprint
376 bytes FRAM

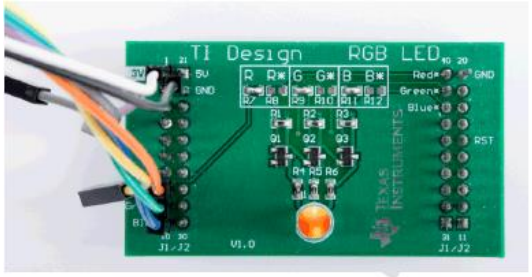
Performance

12 UART selectable colors available
9600 baud UART interface

Applications

User Interface/HMI Systems





TIDM-G2XXSWRGBLED

Tech Doc Code Example

Stepper Motor Control

Overview

1/16th micro-stepping mode with UART interface controls the frequency from 0 → 250KHz. Timer_B of the low cost MSP430FR2000 MCU is used to vary the frequency and maintain a 50% duty cycle.

Resources

Hardware Peripherals
- Timer_B
- eUSCI_A (UART mode)

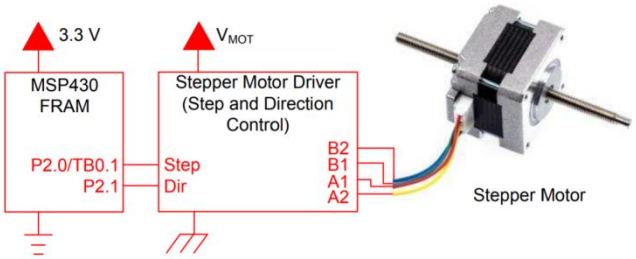
Firmware Footprint
~250 bytes FRAM

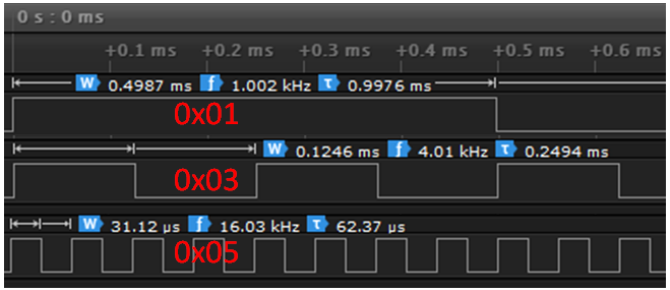
Performance

LPM0 used in idle mode with ~300uA standby current

Applications

Robotics, Industrial Automation, Device Positioning





PWM Frequency Modulation

Tech Doc Code Example

Servo Motor Controller

Overview

This implementation uses the low cost MSP430FR2000 device in conjunction with a driver MOSFET to produce a variable pulse width of 1 → 2ms with 16 overall control steps.

Resources

Hardware Peripherals
- Timer_B
- eUSCI_A (UART mode)

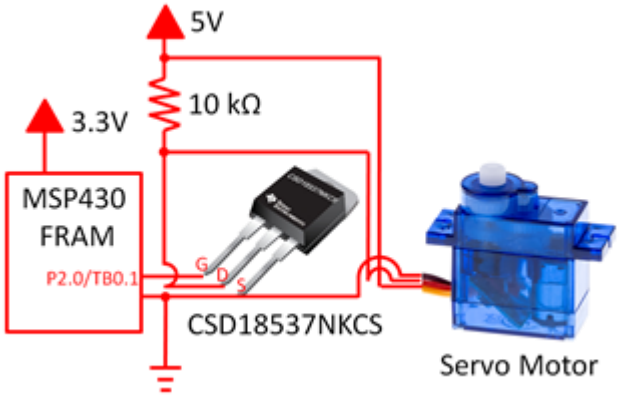
Firmware Footprint
~200 bytes FRAM

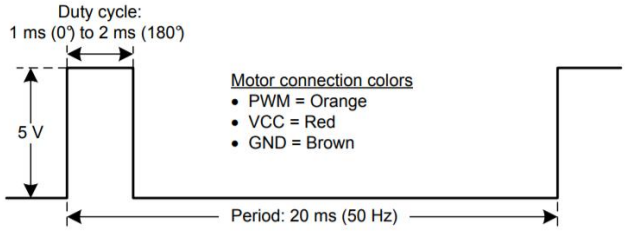
Performance

16 Steps from 1ms → 2ms relating to 0 → 180deg of motion

Applications

Robotics, Industrial Automation, Device Positioning





Servo Motor Connections and Waveform

Tech Doc Code Example

Dual-Output 8-Bit PWM DAC

Overview

This design shows the implementation of two 8-bit PWM DAC's using two Timer_B channels using MSP430FR2000 MCU. One outputs a 250Hz sine wave and the other outputs a constant DC level. These outputs can be easily modified by adjusting lookup table values.

Resources

Hardware Peripherals
- Timer_B

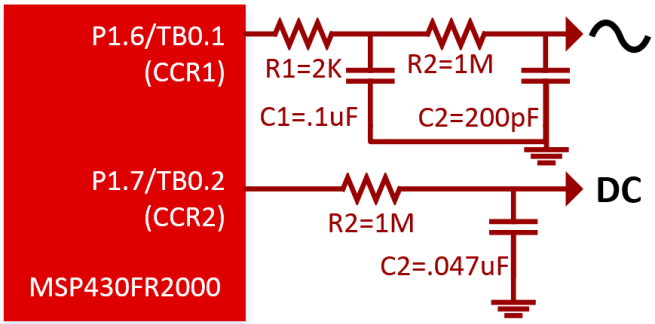
Firmware Footprint
232 bytes FRAM

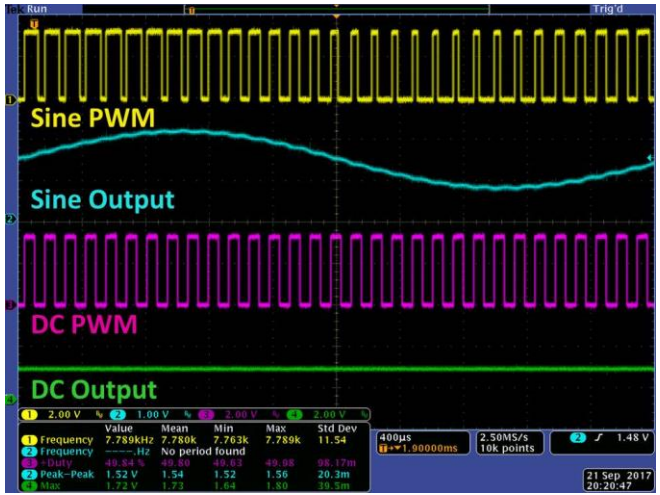
Performance

8-bit, 250Hz Sine Wave Output.

Applications

Musical Tuners, Alarms, Toys & Games





Tech Doc Code Example

Analog Input to PWM Output Using the Enhanced Comparator

Overview

An ADC converts an analog signal, such as voltage or current to a digital number proportional to its magnitude. In the absence of an on-chip ADC on the MSP430FR2000 MCU, a low-cost, low-accuracy comparator can be used. Also, in some applications, the output must be translated into a pulse width modulated signal.

Resources

Hardware Peripherals
- eComp
- GPIO
- Timer_B

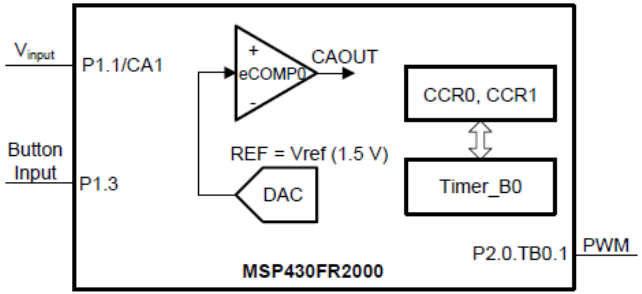
Firmware Footprint
252 bytes FRAM

Performance

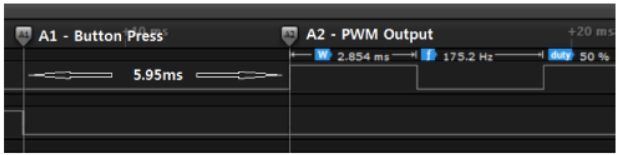
6 Bit Resolution

Applications

DC Electric Motor Control



eCOMP ADC and PWM Diagram



Analog-to-PWM

Tech Doc Code Example

Timer Functions

External RTC with Backup Memory

Overview

Based around the low cost MSP430FR2000 MCU, using the POSIX time format, this software RTC implementation offers a simple solution for basic real time clock needs. 16 bytes of ultra low power non volatile storage are also provided.

Resources

Hardware Peripherals
- RTC Counter
- eUSCI_A (UART mode)

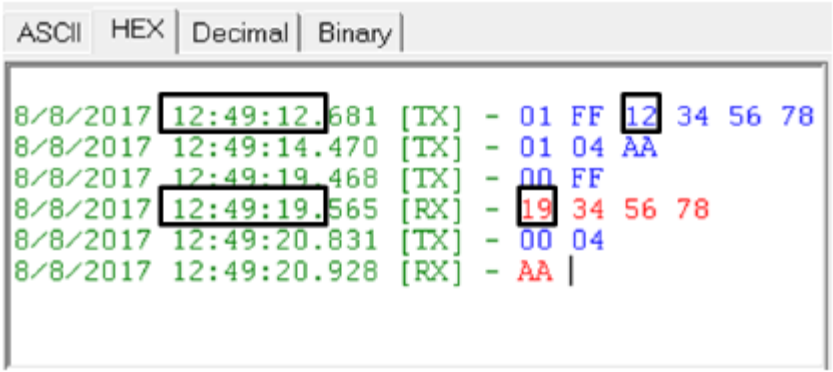
Firmware Footprint
376 bytes FRAM

Performance

Timestamp is stored in FRAM and is updated every second

Applications

Fire & Security Systems



RTC and Backup Memory

Tech Doc Code Example

Simple RTC-Based System Wake-Up Controller

Overview

This fixed interval system wake up controller offers ≤1.5uA operation. Based on the low cost MSP430FR2000 MCU, this solution can easily be used to wake up a less energy efficient application processor or wireless interface.

Resources

Hardware Peripherals
- RTC Counter
- GPIO

Firmware Footprint
246 bytes FRAM

Performance

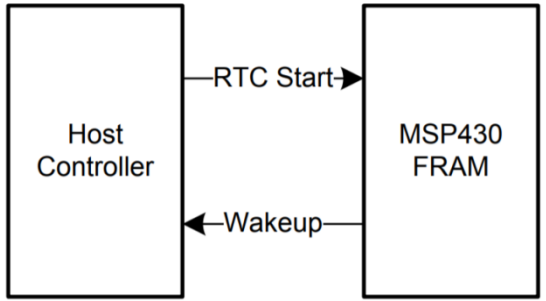
Average power consumption ~1.5uA with 1 second wake-up time

Applications

Wireless Sensor Nodes, Security Systems, Battery Powered Meters

Wake-up Time:	1 s	10 s	1 m*	1 h*	24 h*
Avgerage Current:	1.5 µA	1.06 µA	1.00 µA	<1.00 µA	<1.00 µA

Average Power Consumption



Tech Doc Code Example

Programmable System Wake-Up Controller

Overview

This programmable interval wake up controller offers <1.5uA average operation. Based on the low cost MSP430FR2000 MCU, the wake up timing can be modified via the SPI interface from an external host processor or other higher power controller.

Resources

Hardware Peripherals
- RTC Counter
- eUSCI_A (SPI mode)
- GPIO

Firmware Footprint
364 bytes FRAM

Performance

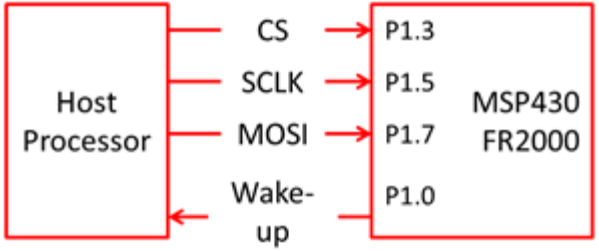
Average power consumption ~1.3uA

Applications

Wireless Sensor Nodes, Security Systems, Battery Powered Meters

Average Power Consumption (* Estimated)

Wake-Up Time	1s	10s	1 min*	1hr*	24 hr*
Avg Current (uA)	1.36	1.33	1.32	1.32	1.32



Tech Doc Code Example

External Programmable Watchdog Timer

Overview

An external watchdog timer can add an additional level of redundancy to embedded systems. This low cost MSP430FR2000 MCU-based implementation allows the user to change the watchdog timer (WDT) period from 1 → 2s with simple UART commands.

Resources

Hardware Peripherals
- Timer_B
- eUSCI_A (UART mode)

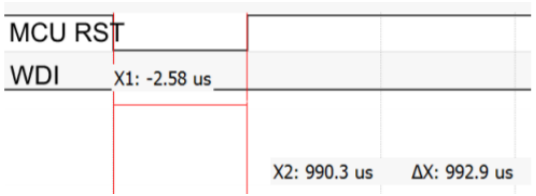
Firmware Footprint
374 bytes FRAM

Performance

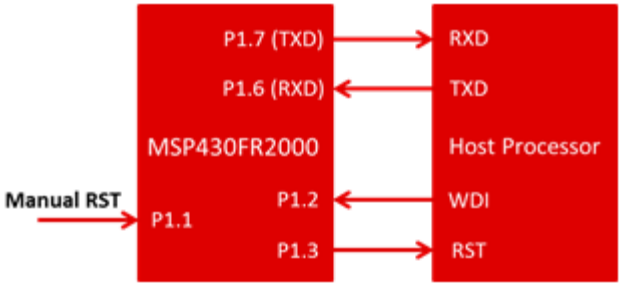
1 – 2 second WDT timeout period
200ms steps via UART control

Applications

Smoke Detectors, Other Critical Systems



Invalid WDI and Reset



Tech Doc Code Example

7-Segment LED Stopwatch

Overview

Many applications require the need to keep time using simple low-power methods. This implementation demonstrates the MSP430FR2000 MCU's RTC module and outputs a 3 digit timing parameter to an external 3-digit 7 segment LED display.

Resources

Hardware Peripherals
- RTC Counter
- GPIO

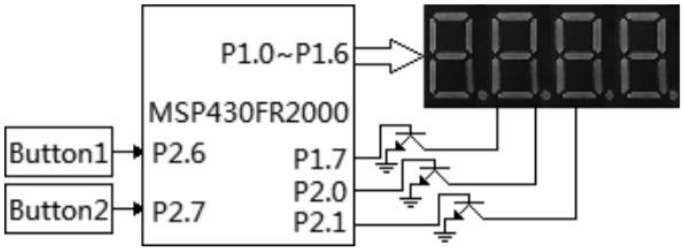
Firmware Footprint
358 bytes FRAM

Performance

Counts from 0.1 seconds to 99.9 seconds and can be scaled to 1 second to 999 seconds

Applications

Kitchen Appliances, Stopwatch, Time-keeping Devices



Stopwatch Block Diagram

Active LED Display

Tech Doc Code Example

Voltage Monitor With a Timestamp

Overview

Voltage monitoring is essential for battery and bus powered applications to save the system state through a power loss event. Often the real time occurrence should be saved as a timestamp to track the power loss event. The MSP430FR2000 device implements these procedures.

Resources

Hardware Peripherals
- eComp
- RTC Counter
- eUSCI_A (UART mode)

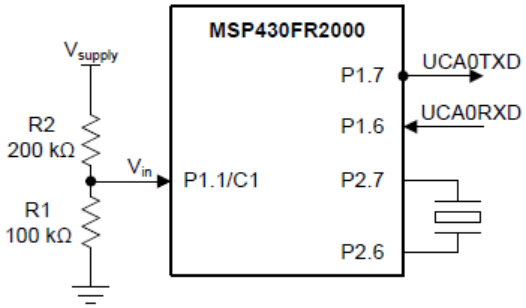
Firmware Footprint
368 bytes FRAM

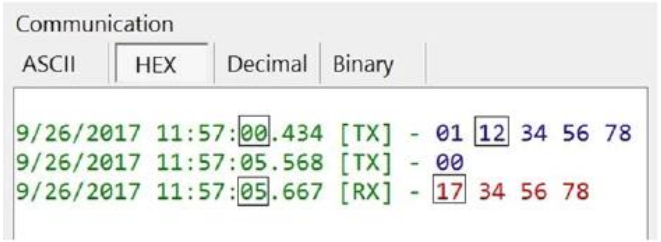
Performance

30-mV hysteresis mode to avoid false triggers

Applications

Data Logging Systems





Write and Read RTC Example

Tech Doc Code Example

System Functions

EEPROM Emulation

Overview

The MSP430FR2000 device can function as a serial EEPROM replacement. Using a small code footprint, the device can provide 48 bytes of EEPROM functionality.

Resources

Hardware Peripherals
- eUSCI_A (SPI mode)
- GPIO

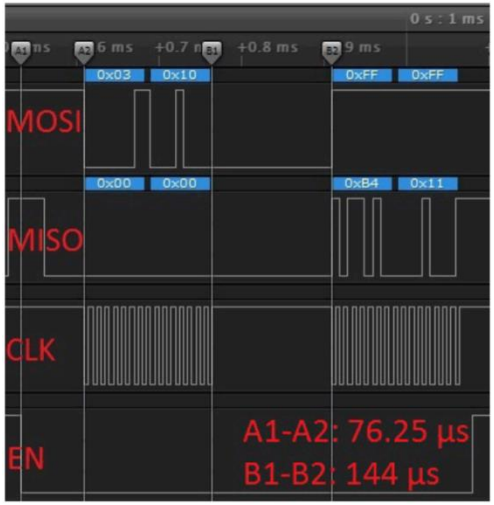
Firmware Footprint
384 bytes FRAM

Performance

20uA @ 3V when in LPM3 mode (excluding SPI data transfers)

Applications

Unit Identification, Backup Information



EEPROM Read Word Sequence

Tech Doc Code Example

Low-Power Hex Keypad

Overview

The MSP430FR2000 device can be implemented in a low power, 16 button keypad system.

Resources

Hardware Peripherals
- eUSCI_A (UART mode)
- Watchdog Timer

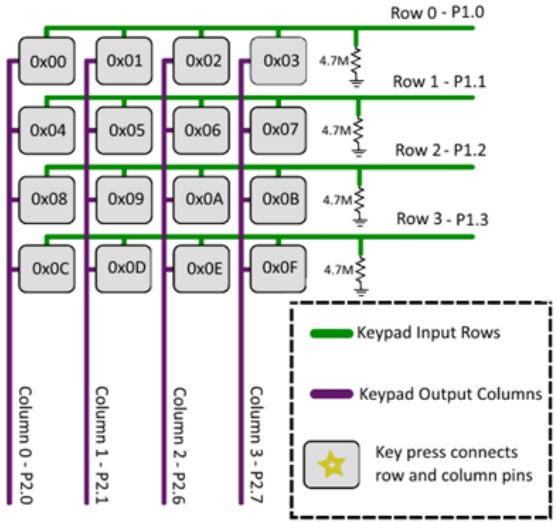
Firmware Footprint
342 bytes FRAM

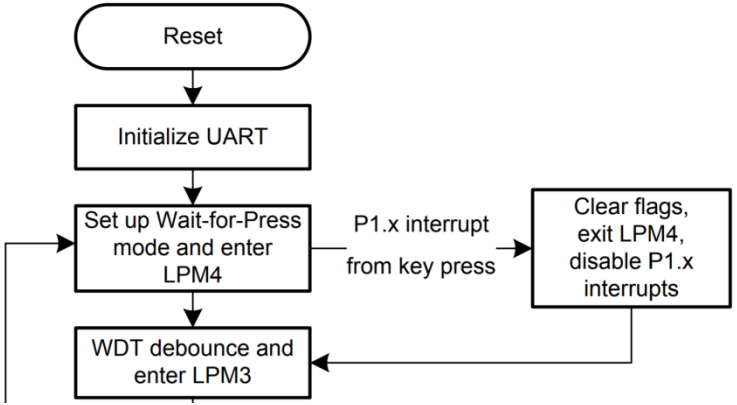
Performance

0.58uA in standby mode waiting for a key press

Applications

Security Systems, Electronic Smart Locks





Tech Doc Code Example

Single-Slope Analog-to-Digital Conversion Technique

Overview

A slope converter is an easy way to add ADC functionality to an MCU only featuring a comparator module. This example shows a low power, high accuracy implementation based around the MSP430FR2000 device .

Resources

Hardware Peripherals
- eComp
- Timer_B

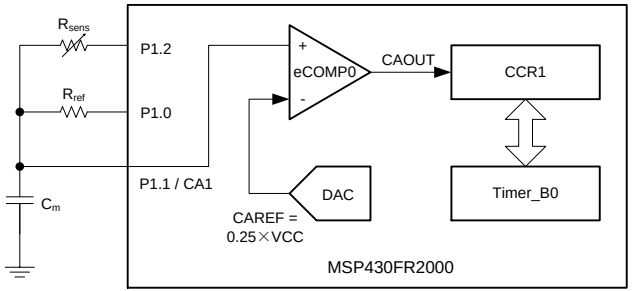
Firmware Footprint
384 bytes FRAM

Performance

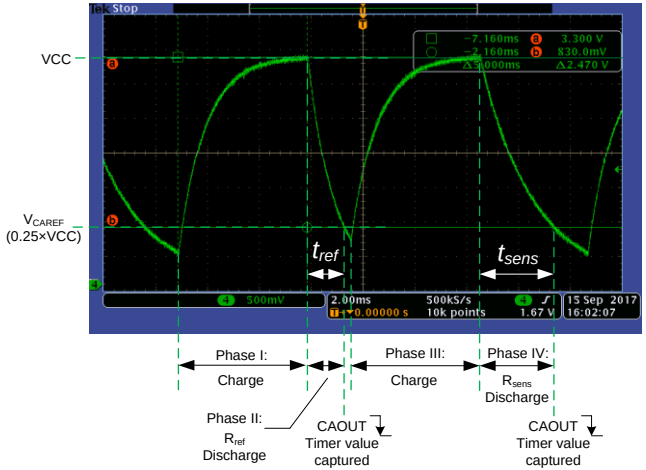
0.03% measurement accuracy of 12K resistor @ 75Hz.

Applications

Low-Cost Multi-Meters, Thermostats



Voltage at Cm During Resistance Measurement



Tech Doc Code Example

Multi-Function Reset Controller

Overview

Host processors may require both a HW reset or a SW reset. The MSP430FR2000 MCU offers a simple solution to address both types of resets using a single button.

Resources

Hardware Peripherals
- Timer_B

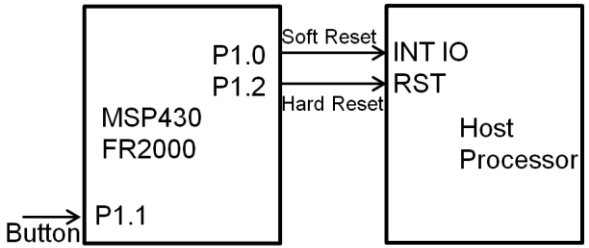
Firmware Footprint
310 bytes FRAM

Performance

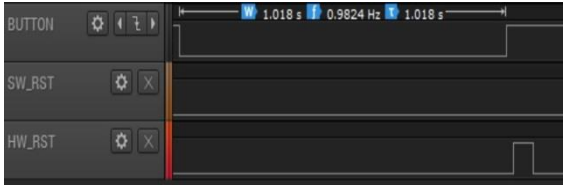
Button press <0.5 seconds triggers soft reset
Button press >1 second triggers hard reset

Applications

Computers, Smart Phones



Soft Reset Pulse



Hard Reset Pulse

Tech Doc Code Example

Hysteresis Comparator with UART

Overview

Comparators are used to differentiate between two different signal levels. Because signal variation at the comparison threshold can cause multiple transitions, hysteresis upper and lower limits are applied to minimize the effects of noise. A solution can be implemented with the MSP430FR2000 device.

Resources

Hardware Peripherals
- eComp

Firmware Footprint
292 bytes FRAM

Performance

7uA with reference voltage of 3.3V

Applications

Analog Sensors, Switching Power Supplies, Level Detectors

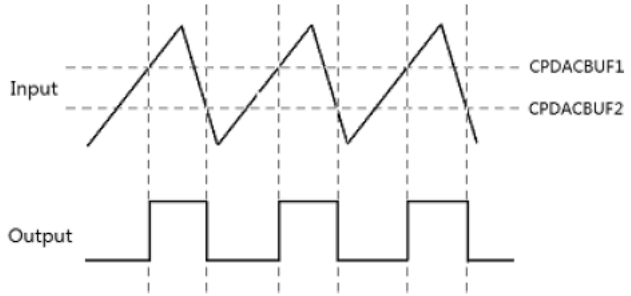
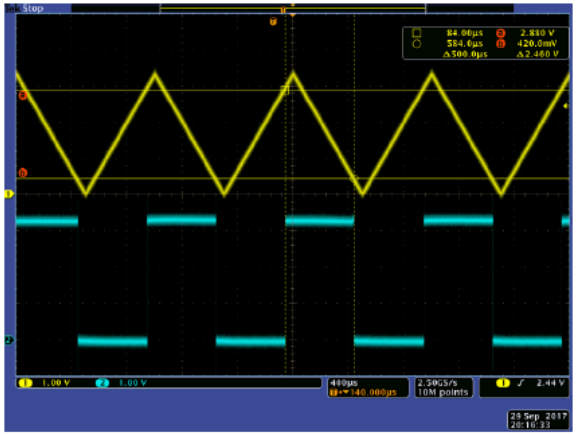


Figure 1



Scope of Input and Hysteresis Comparator Output

Tech Doc Code Example

ADC Wake and Transmit on Threshold

Overview

Many applications require sampling analog signals periodically so action can be taken based on the behavior of those signals. ADCs can be triggered with precise timers to provide a solution to this requirement. This design utilizes the MSP430FR2100 MCU.

Resources

Hardware Peripherals
- ADC10
- eUSCI_A (UART mode), GPIO
- Timer_B

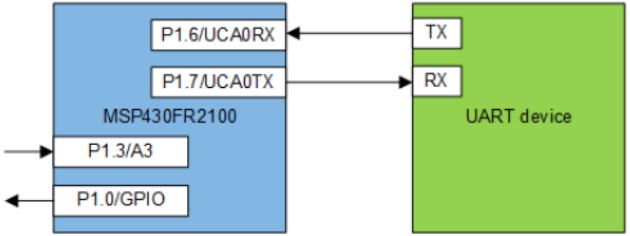
Firmware Footprint
310 bytes FRAM

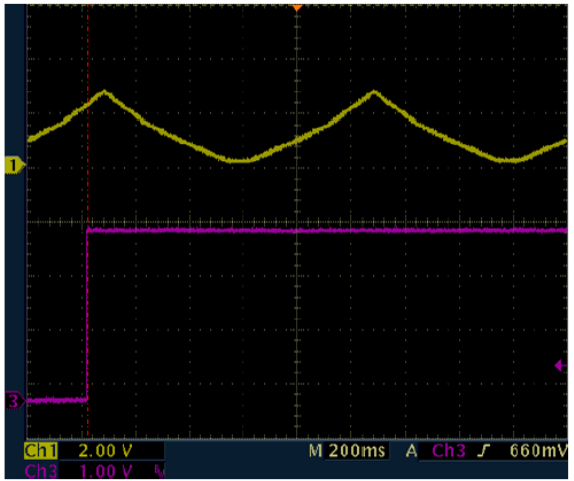
Performance

25uA standby current at 3V

Applications

Battery Monitors, Thermostats





Scope of Triangular Input and GPIO Output

Tech Doc Code Example

Tamper Detection

Overview

This design focuses on detecting when a secured enclosure is opened. The described system uses the MSP430FR2000 device, and provides tamper detection and evidence. This is typically used as the first line of defense in system applications.

Resources

Hardware Peripherals
- RTC Counter
- GPIO

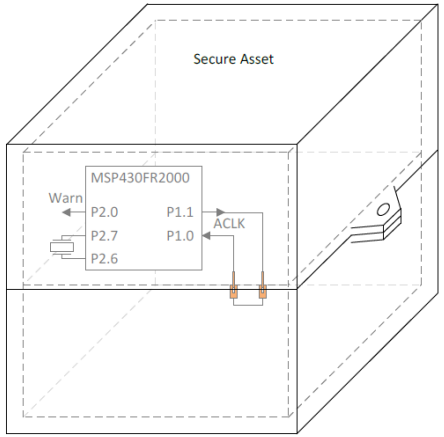
Firmware Footprint
< 512 bytes FRAM

Performance

Tamper alert detected in <490uS

Applications

Utility Meters, Electronic Door Locks, Door/Window Sensors



Hardware Connection Diagram

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1D 4B DE C0 B2 40 80 5A CC 01 C2 43 04 20 E2 D3
04 02 E2 D3 02 02 D2 D3 02 02 D2 D3 06 02 D2 D3
05 02 D2 C3 03 02 B2 40 10 A5 A0 01 32 D0 40 00
```

Protected Data Stored in FRAM

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF B2 40 80 5A CC 01 C2 43 04 20 E2 D3
04 02 E2 D3 02 02 D2 D3 02 02 D2 D3 06 02 D2 D3
05 02 D2 C3 03 02 B2 40 10 A5 A0 01 32 D0 40 00
```

Protected Data Erased from FRAM

Tech Doc Code Example

Quadrature Encoder Position Counter

Overview

Quadrature encoders are used to keep track of the angular position of knobs and motors in many applications including volume control, robotics and factory automation systems. The MSP430FR2000 device is implemented in this system.

Resources

Hardware Peripherals
- eUSCI_A (UART mode)
- RTC Counter

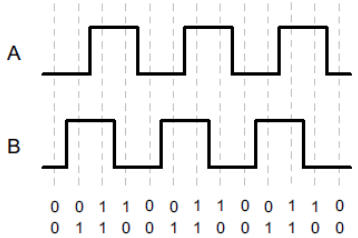
Firmware Footprint
< 512 bytes FRAM

Performance

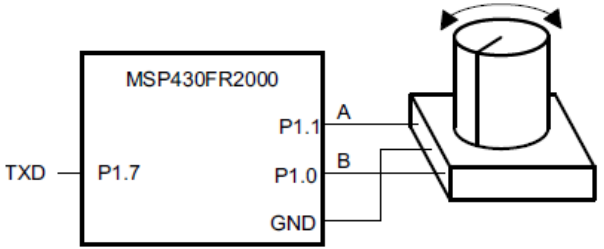
Track angular velocity up to 150,000 RPM

Applications

HMI Knobs or Wheels, Robotics



Quadrature Encoder Clockwise Rotation Waveforms



Tech Doc Code Example

Programmable Clock Source

Overview

Utilizing the low cost MSP430FR2000 MCU, this clock source provides a UART or SPI selectable frequency output of 1MHz, 2MHz, 4MHz, 8MHz or 16MHz.

Resources

Hardware Peripherals
- eUSCI_A (SPI/UART mode)
- GPIO

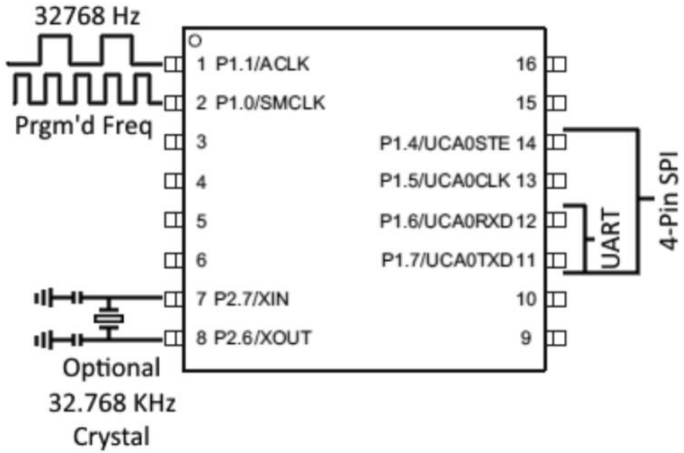
Firmware Footprint
344 bytes FRAM

Performance

Typical power consumption ~400uA @ 3V operating at 1 – 16MHz.

Applications

Low-frequency RF systems



Tech Doc Code Example

Programmable Frequency Locked Loop

Overview

This MSP430FR2100 MCU based variable clock generator utilizes the frequency locked loop to provide an output frequency between 0 and 16MHz. The frequency can be controlled by using simple UART or SPI commands.

Resources

Hardware Peripherals
- eUSCI_A (SPI/UART mode)
- GPIO

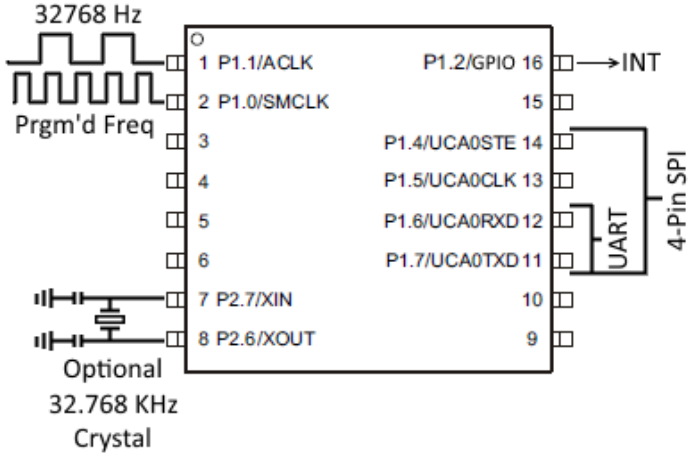
Firmware Footprint
872 bytes FRAM

Performance

Typical power consumption ~400uA @ 3V with outputs between 0 – 16MHz.

Applications

Low-frequency RF Systems, Modulation Transceivers



Tech Doc Code Example

Communication Functions

Single-Wire Communication Host

Overview

Many applications utilize a bi-directional single wire interface to connect to a range of IC's, such as temperature sensors, gas gauges, EEPROM's etc. This MSP430FR2000 MCU implementation shows how host functionality is achieved under 512 bytes.

Resources

Hardware Peripherals
- GPIO

Firmware Footprint
384 bytes FRAM

Performance

~1.2mA when communicating
~18uA when idle

Applications

Temperature Sensors, Battery Monitors

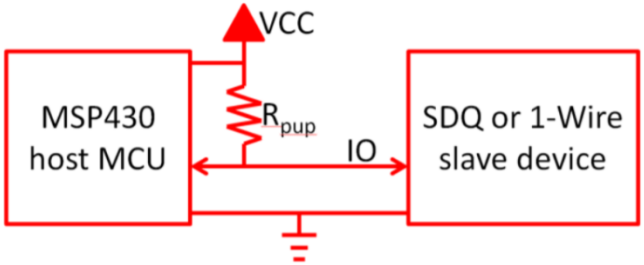
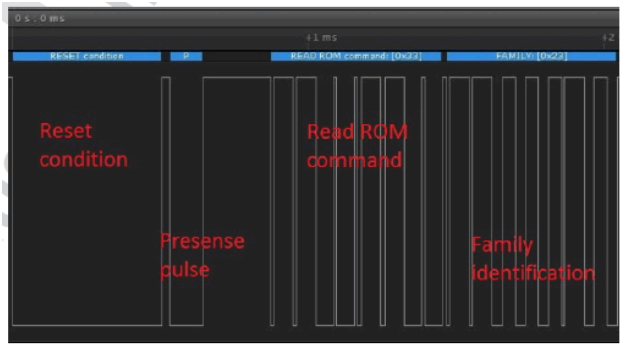


Figure 1



Single-Wire Reset Pulse, Presence Pulse, and ROM Command

Tech Doc Code Example

UART-to-UART Bridge with Baud Rate Translation

Overview

Half Duplex UART to UART bridge designed to interface 2 UART's operating at different BAUD rates. Based on the low cost MSP430FR2000 MCU, this utilizes both a hardware and software UART implementation.

Resources

Hardware Peripherals
- Timer_B
- eUSCI_A (UART mode)
- GPIO

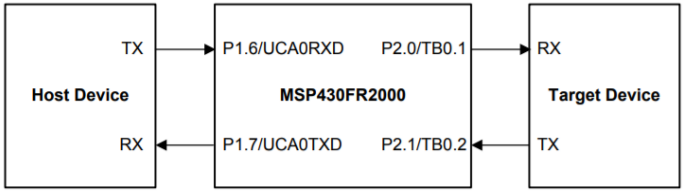
Firmware Footprint
382 bytes FRAM

Performance

Hardware UART up to 921600 bps
Software UART up to 38400 bps

Applications

Protocol Converters, Communication Speed Bridges



UART-to-UART Bridge Diagram

Interface	SMCLK	Max Baud Rate
Hardware UART	16 MHz	921600 bps
Software UART	16 MHz	38400 bps

Tech Doc Code Example

UART-to-SPI Bridge

Overview

A UART to SPI communication bridge can be a useful function when interfacing two IC's or systems together. This example implements a HW SPI master and a SW UART interface with the MSP430FR2000 device.

Resources

Hardware Peripherals
- eUSCI_A (SPI mode)
- Timer_B

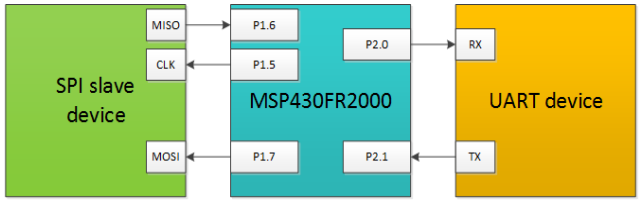
Firmware Footprint
376 bytes FRAM

Performance

SPI Max : 8MHz bit clock
UART Max : 38,400 bps BAUD

Applications

Protocol Converters



UART-to-SPI Diagram

Interface	Max Rate
SPI	8 MHz bit clock
SW UART	38400 bps BAUD rate

Tech Doc Code Example

SPI I/O Expander

Overview

I/O expanders can be used to offload low priority tasks such as LED blinking and button monitoring from a host processor. This simple implementation with the MSP430FR2000 device shows a SPI to 8-pin GPIO expander.

Resources

Hardware Peripherals
- eUSCI_A (SPI mode)
- GPIO

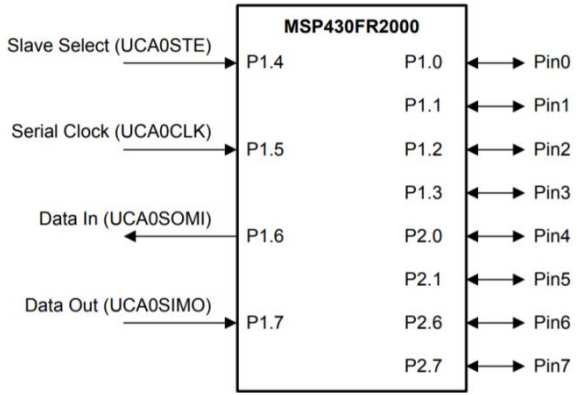
Firmware Footprint
360 bytes FRAM

Performance

Read Response Time: 0.359ms
Write Response Time: 0.295ms

Applications

Parallel to Serial Converters, I/O Expanders, LED Lighting



SPI I/O Expander Diagram

	D7	D6	D5	D4	D3	D2	D1	D0
Read	1	0	0	0	0	0	0	0
Write-DIR	0	0	0	1	0	0	0	0
Write-OUT	0	0	1	0	0	0	0	0

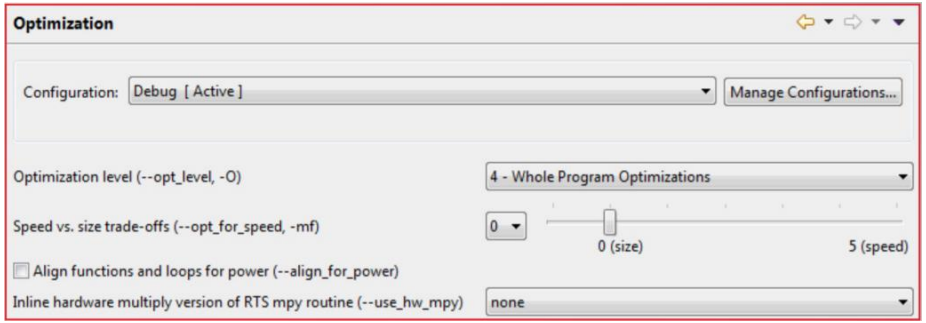
Master Commands

Tech Doc Code Example

Code Optimization

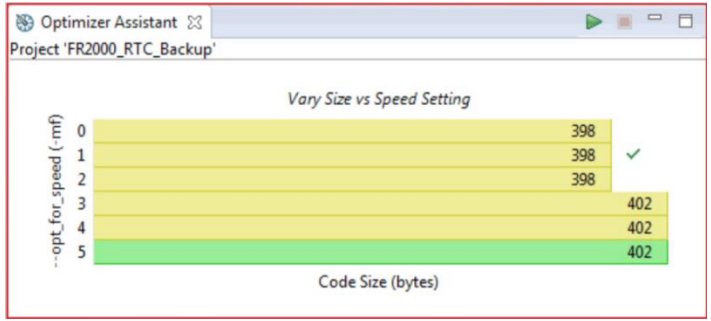
Optimization Settings

- Accessible in Project > Properties > Build > MSP430 Compiler > Optimization
- Optimization Level
  - Determines what types of optimizations the compiler can make
    - Register Optimizations
    - Local Optimizations
    - Global Optimizations
    - Inter-procedure Optimizations
    - Whole Program Optimizations
- Speed vs Size
  - Tells compiler if trade-offs should be made more in favor of size or speed
  - Adjust from 0(size) – 5(speed)



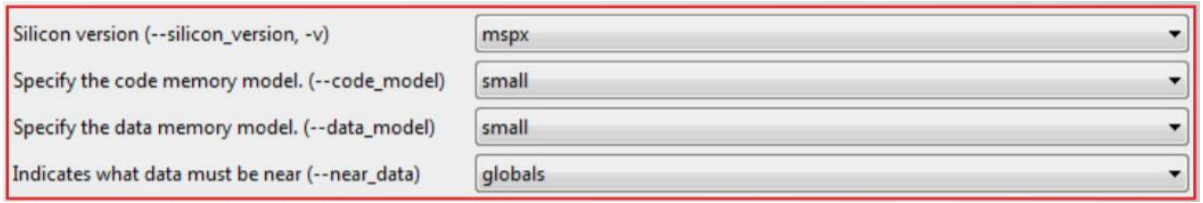
Optimizer Assistant

- Accessible in View > Optimizer Assistant
- Allows multiple builds of project with varied optimization settings
- Option to vary speed vs size trade-offs or optimization level
  - Only 1 build option can be varied at a time (speed vs size or optimization level)
  - The build option not being varied will stay constant at selected setting while the other build option is varied
- Optimizer will display code sizes for different settings
  - Red indicates the code cannot fit in the selected device
  - Yellow indicates the code fits in the device, but there is a better option available
  - Green indicates the recommended option for optimal speed while still fitting into the device
  - Check mark indicates the current selection in the project settings



Code and Data Model

- Accessible in Project > Properties > Build > MSP430 Compiler > Processor Options
- Larger MSP430 devices have code space that extends to addresses of 0x10000 and above
  - Extended instruction set is needed to support operations in these 20-bit addresses
- We should ensure that only the base 16-bit instruction set is used for devices where no addresses above 0x10000 exist
- This is controlled by selecting the correct code and data model in the IDE project settings
  - Select Small Code Memory Model and Small Data Memory Model



Controlling Global Variable Initialization

- C-initialization code for global variables may take up more space than directly initializing global variables with user code
- In order to control global variable initialization:
  - Move initialization of global variables into main().

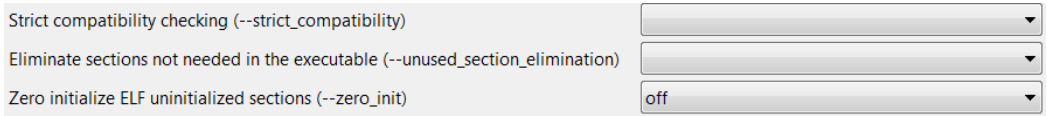
```
#include <msp430.h>

//unsigned char RXData = 0;
//unsigned char TXData = 1;
unsigned char RXData;
unsigned char TXData;

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                // Stop watchdog timer

    //Initialize globals
    RXData = 0;
    TXData = 1;
```

  - Disable zero initialization in CCS by navigating to Project > Properties > MSP430Linker > Advanced Options > Miscellaneous and set the Zero initialize ELF uninitialized sections to “off”



Coding Techniques

- Use Smallest Possible Types for Variables and Constants
  - Use 8-bit types vs. 16-bit types when possible
  - When using the PERSISTENT keyword, use the smallest type of variable possible to minimize FRAM space
  - Minimize lookup table values when possible
    - eg: For a PWM lookup table, implement timer dividers to decrease lookup table values
- Avoid Multiply and Divide
  - Use bit shifts for multiplication or division by powers of 2
  - Utilize lookup tables when possible
- Use Word Access to CPU Registers
  - Initialize the entire port with one write

```
PAOUT = BIT15 | BIT0;                vs    P1OUT = BIT0; P2OUT = BIT7;
```

- Write to CPU Registers Only Once (Where Possible)

```
TA0CCTL2 = OUTMOD_4 | CCIE;          vs    TA0CCTL2 |= CCIE;
                                           TA0CCTL2 &= ~OUTMOD_7;
                                           TA0CCTL2 |= OUTMOD_4;
```

Note: when using =, any other bits in register that need to remain set should be written

Coding Techniques

- Use intrinsics that will enable compiler to make optimizations during code generation
  - __even_in_range(x,NUM);
    - Tells compiler value x will always be an even value between 0 to NUM inclusive. This information allows the compiler to further optimize code than it normally would.

```
switch(__even_in_range(UCA0IV,USCI_UART_UCTXCPTIFG))
{
    case USCI_NONE: break;
    case USCI_UART_UCRXIFG:
          UCA0IFG &=~ UCRXIFG;            // Clear interrupt
          RXData = UCA0RXBUF;             // Clear buffer
          if(RXData != TXData)            // Check value
          {
              P1OUT |= BIT0;              // If incorrect turn on P1.0
              while(1);                   // trap CPU
          }
          TXData++;                                 // increment data byte
          __bic_SR_register_on_exit(LPM0_bits); // Exit LPM0 on reti
          break;
    case USCI_UART_UCTXIFG: break;
    case USCI_UART_UCSTTIFG: break;
    case USCI_UART_UCTXCPTIFG: break;
}
```

  - __low_power_mode_x(void);
    - Enables a low power mode of user's choice (x)

Resources

- 25 Functions for 25 Cents Blog (Includes Links to TechNotes)
- eBook (Includes Porting from FR2311 to FR2000 Information)
- Code Examples
- Video Series

Additional Resources

- MSP430 Value Line Sensing
- MSP430FR4xx_2xx User's Guide
- Code Porting From MSP430FR2000 to MSP430FR2311 MCUs
- Optimizing C Code for Size With MSP430™ MCUs: Tips and Tricks
- Migrating From the MSP430F2xx and MSP430G2xx Families to the MSP430FR4xx/FR2xx Family
- How to Use the Integrated Operational Amplifiers on MSP430FR2311
- VLO calibration on the MSP430FR4xx AND MSP430FR2xx
- Software I2C master and slave on MSP430FR2111
- General Oversampling of MSP ADCs for Higher resolution
- Software UART (C example code is also available)
- Designing With MSP430™ MCUs and Segment LCDs