TI Designs

Memory Emulation Using 1-Wire® Communication Protocol

TEXAS INSTRUMENTS

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDM-1WIREFREEPROM	Design Folder
MSP-EXP430FR5969	Product Folder

TI E2E™ Community

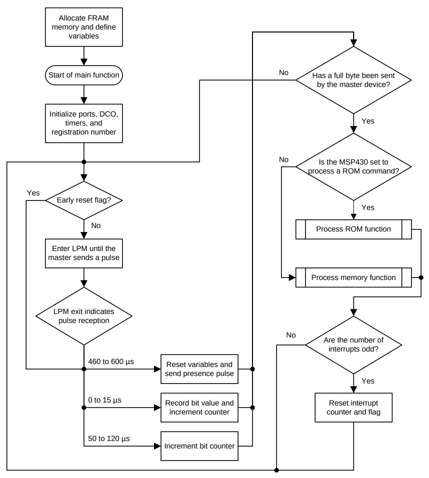
ASK Our E2E Experts
WEBENCH® Calculator Tools

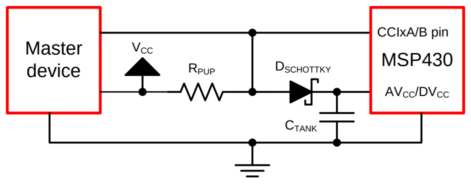
Design Features

- Complete 1-Wire® Protocol Support Assures an Easy Transition and Effortless Substitution for Similar EEPROM Devices
- Uses LPMs to Accomplish an Average Standby Current Consumption of 600 nA in Standby Mode.
- Design is Achievable On Any MSP430™ FRAM Device and Allows for Flexibility in the Available EEPROM Memory Space
- Includes Selectable Operation Modes Such as Dedicated or Parasitic Power
- Provided Firmware May be Modified for Use In Other 1-Wire Applications

Featured Applications

- Additional Device Memory Storage
- Device Identification and Authentication
- Personal Electronics Peripherals


Allocate FRAM memory and define variables
Start of main function
Initialize ports, DCO, timers, and registration number
Early reset flag?
Yes
No
Enter LPM until the master sends a pulse
LPM exit indicates pulse reception
460 to 600 µs
Reset variables and send presence pulse
0 to 15 µs
Record bit value and increment counter
50 to 120 µs
Increment bit counter
Has a full byte been sent by the master device?
No
Yes
Is the MSP430 set to process a ROM command?
No
Yes
Process ROM function
Process memory function
Are the number of interrupts odd?
No
Yes
Reset interrupt counter and flag



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

The TIDM-1WIREEEPROM is a firmware-based reference design that emulates EEPROM functionality on an MSP430 ferroelectric RAM (FRAM) device which is compatible with the 1-Wire® communication protocol True to its name, the 1-Wire achieves communication between the master and slave devices through a common line that is pulled up by a resistor. By following the 1-Wire protocol, which dictates the speed that the communication must occur, the slave devices influence the polarity of the data line as it is polled by the master. 1-Wire slave devices successfully receive commands and send responses using a single wire. Another unique advantage of this bus system is that it only needs two wires, data and ground, connected between devices with power being sourced directly through the data line. This operating mode is called parasitic power (as compared to the typical dedicated power). To attain parasitic power, insert a Schottky diode and tank capacitor to power the slave device when the data line is actively being pulled low.

Through the application of Low Power Modes (LPMs) and FRAM technology, replicating 1-Wire EEPROM capability is possible on an MSP430 device. The replication of the 1-Wire EEPROM capability on an MSP430 device has been tested with an MSP-EXP430FR5969 LaunchPad™ and TM4C129XNCZAD development board, which is used as the 1-Wire master device. Operating inside of LPMs in an efficient manner lets the MSP430FR5969 device employ parasitic power mode or remain with dedicated power mode. The MCU enters a more economical LPM during standby mode where no communication occurs, resulting in the lowest system current consumption possible. The flexibility in allocating FRAM space for EEPROM memory also allows this solution more flexibility and data storage as compared to other available 1-Wire devices.

This TI Design contains the resources to imitate 1-Wire EEPROM functionality on an MSP430 FRAM device. This design includes an explanation of both the simple hardware connections needed, and the provided Code Composer Studio™ C-code application firmware. Section 3 and Section 4 explain how to switch between dedicated and parasitic power in both the hardware and software, along with allocating FRAM space for EEPROM memory use. The instructions also help the user to understand the purpose and application of each ROM and memory function command under 1-Wire protocol, as well as how to initiate communication with a target 1-Wire master device.

A Saleae logic analyzer was used to test and confirm proper 1-Wire communication protocol between the 1-Wire master and MSP430 FRAM slave device. Several screen shots of the logic analyzer readings have been provided in Section 5 of this guide. Power consumption measurements were taken with a Keysight N670 5B DC power analyzer. CPU and peripheral states were also recorded using EnergyTrace++™ technology software.

1.1 MSP-EXP430FR5969

The MSP-EXP430FR5969 (also called the FR5969 LaunchPad™) is an inexpensive evaluation module for evaluating and applying MSP430 FRAM technology. Beginning development is simple with an onboard eZ-FET emulator used for programming and debugging the MSP430 device, providing back-channel UART through USB to PC, and communicating with EnergyTrace++. Two user buttons and LEDs are incorporated for user interaction without the need for additional hardware. This LaunchPad showcases the MSP430FR5969 16-bit MCU with 64 KB of FRAM, 2 KB of SRAM, CPU speed up to 16 MHz, 5 timer blocks, a 16-channel 12-bit analog-to-digital converter (ADC), and additional digital peripherals including AES256, CRC, MA, and HW MPY32. A 0.1F SuperCap grants the option for stand-alone power, and rapid prototyping is accomplished with the use of 20-pin BoosterPack expansion headers along with a wide range of available plug-in modules encompassed in the BoosterPack ecosystem.

The only peripherals used in this reference design are two separate timers. One timer, TA1, is connected to the data line and uses the CCIxA/B functionality to measure pulse widths sent by the master 1-Wire device. This pin must also be able to switch to general-purpose output operation to respond by pulling on the data line. The application firmware supplied with this reference design uses the MSP430FR5969 CCI1A module input signal from TA1 on pin P1.2 to accomplish this task, but in reality the code can be modified to use any CCIxA/B pin. The other timer, TA0, is used for delays and other general clocking mechanisms. The MPU is used to protect the EEPROM memory stored in the FRAM from inadvertent CPU execution, or read and write access.

2 Block Diagrams

2.1 TIDM-1WIREEEPROM System Block Diagram

Figure 1 shows a typical connection between a master device and corresponding MSP430 slave device that replicates 1-Wire EEPROM functionality. The AV_{cc}/DV_{cc} pins may alternatively be tied directly to V_{cc} if dedicated power mode is desired in the place of parasitic power mode.

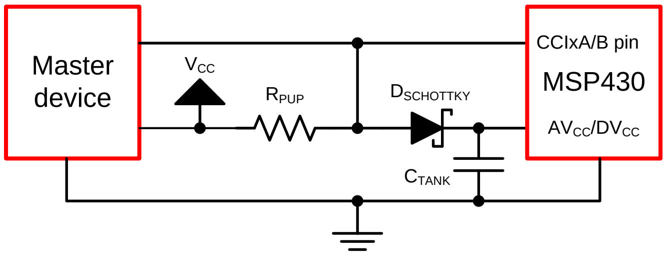


Figure 1. TIDM-1WIREFREEPROM System Block Diagram

2.1.1 MSP-EXP430FR5969 Functional Diagrams

Figure 2 shows the functional diagram of the MSP430FR5869 device. Figure 3 shows the functional diagram for the MSP-EXP430DF5969 LaunchPad board. The upper half of the dotted line represents the eZ-FET, and the bottom half of the dotted line represents the target connections.

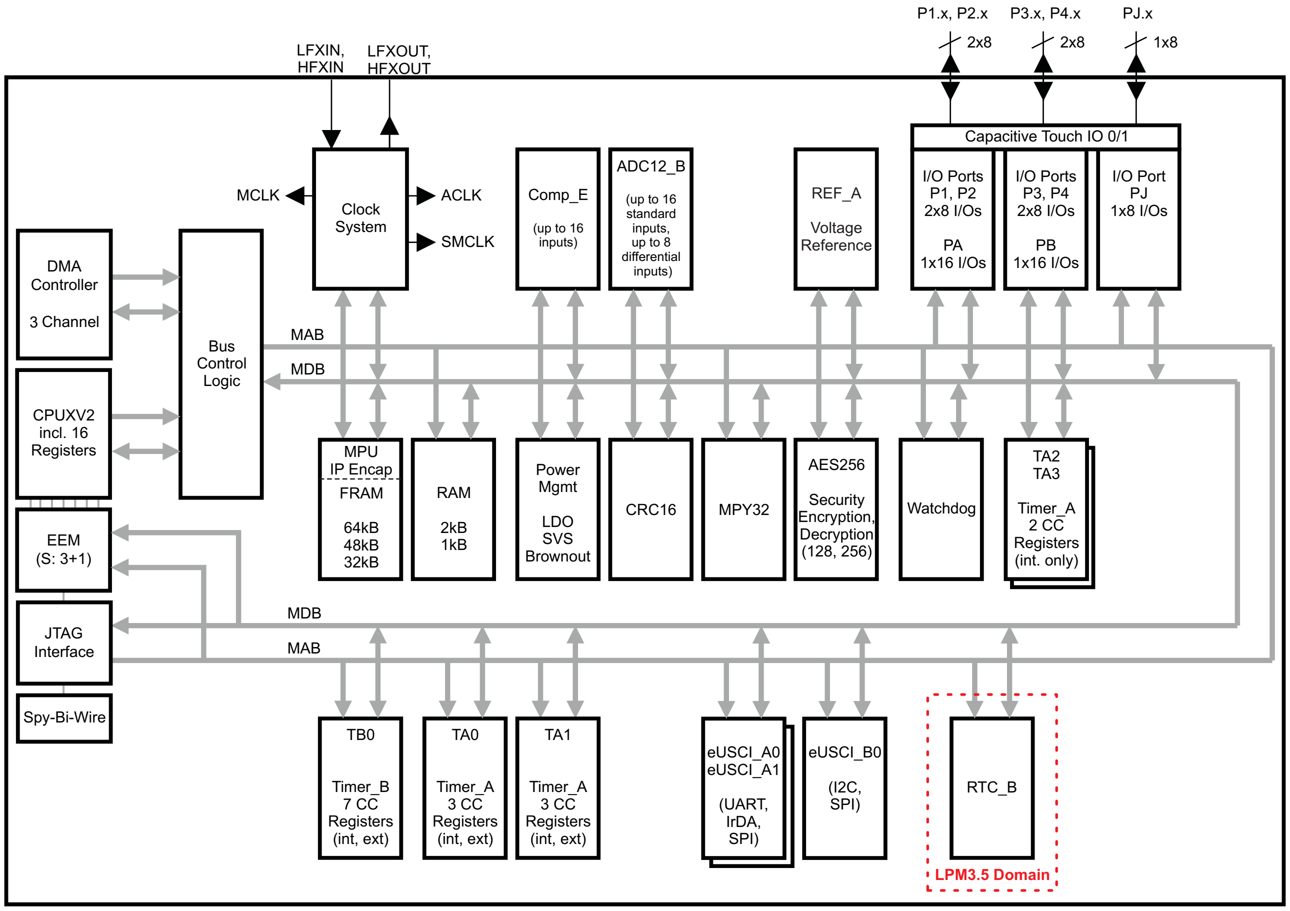

P1.x, P2.x
2x8
P3.x, P4.x
2x8
PJ.x
1x8
LFXIN, HFXIN
LFXOUT, HFXOUT
Capacitive Touch IO 0/1
MCLK
Clock System
ACLK
SMCLK
Comp_E (up to 16 inputs)
ADC12_B (up to 16 standard inputs, up to 8 differential inputs)
REF_A Voltage Reference
I/O Ports P1, P2 2x8 I/Os PA 1x16 I/Os
I/O Ports P3, P4 2x8 I/Os PB 1x16 I/Os
I/O Port PJ 1x8 I/Os
DMA Controller 3 Channel
Bus Control Logic
MAB
MDB
CPUXV2 incl. 16 Registers
MPU IP Encap FRAM 64kB 48kB 32kB
RAM 2kB 1kB
Power Mgmt LDO SVS Brownout
CRC16
MPY32
AES256 Security Encryption, Decryption (128, 256)
Watchdog
TA2 TA3 Timer_A 2 CC Registers (int. only)
EEM (S: 3+1)
JTAG Interface
Spy-Bi-Wire
MDB
MAB
TB0 Timer_B 7 CC Registers (int, ext)
TA0 Timer_A 3 CC Registers (int, ext)
TA1 Timer_A 3 CC Registers (int, ext)
eUSCI_A0 eUSCI_A1 (UART, IrDA, SPI)
eUSCI_B0 (I2C, SPI)
RTC_B
LPM3.5 Domain

Figure 2. MSP430FR5969 Functional Diagram

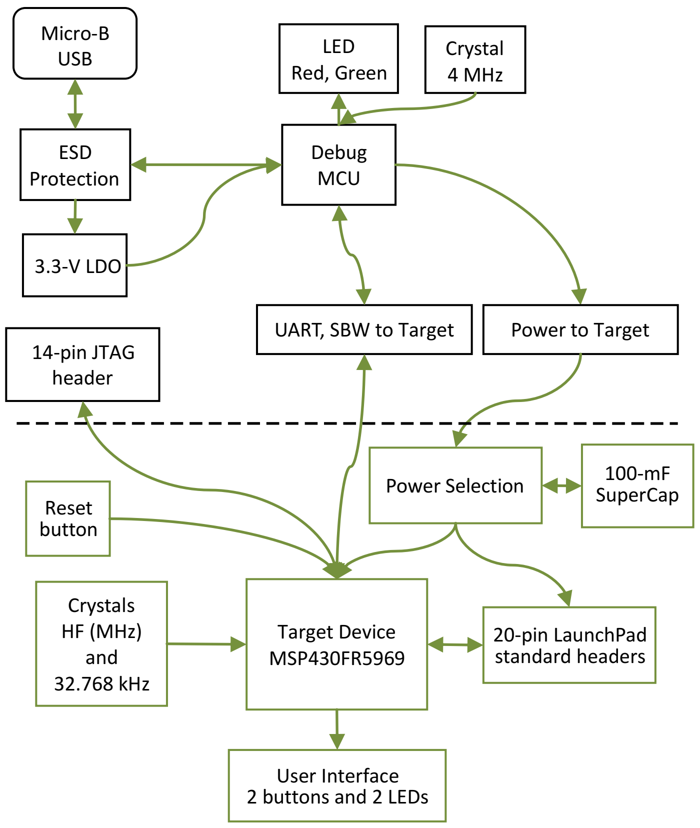


Figure 3. MSP-EXP430FR5969 Functional Diagram

3 Hardware Considerations

3.1 Dedicated Power Operation

Setting up the MSP-EXP430FR5969 LaunchPad hardware for 1-Wire operation in dedicated power mode needs only three connections: power, ground, and data. GND must be common between the MSP430 device and the 1-Wire master device, but the power for each may be shared or individually supplied. The data line must be sourced from an MSP430 device pin that switches between timer CCIxA/B and GPIO functionality. P1.2 with a CCI1A timer module input was chosen as the default pin for this design. Connect the data line to the functional 1-Wire pin of the master and pull it up with a resistor, R_{Pup}, value ranging from 270 to 2200 Ω.

If EnergyTrace readings or the power supply from the MSP-EXP430FR5969 onboard micro-USB connector are desired, the LaunchPad jumpers must be properly oriented to connect the MSP430FR5969 device to the eZ-FET portion of the board. Set J2 to *Bypass* and J10 to *Debugger*. J9 and a minimum of GND, V+, RST, and TST of the J13 jumper bank must be populated. J11 must remain unpopulated at the super capacitor is not investigated or used in this design. If, however, an external power supply is desired, then J10 should be switched to *External*, and all of the jumpers from the J13 bank must be unpopulated. If debugging capabilities are desired while using an external power supply, only GND, RST, and TST of J13 must be populated. Figure 4 shows the placement of the LaunchPad headers.

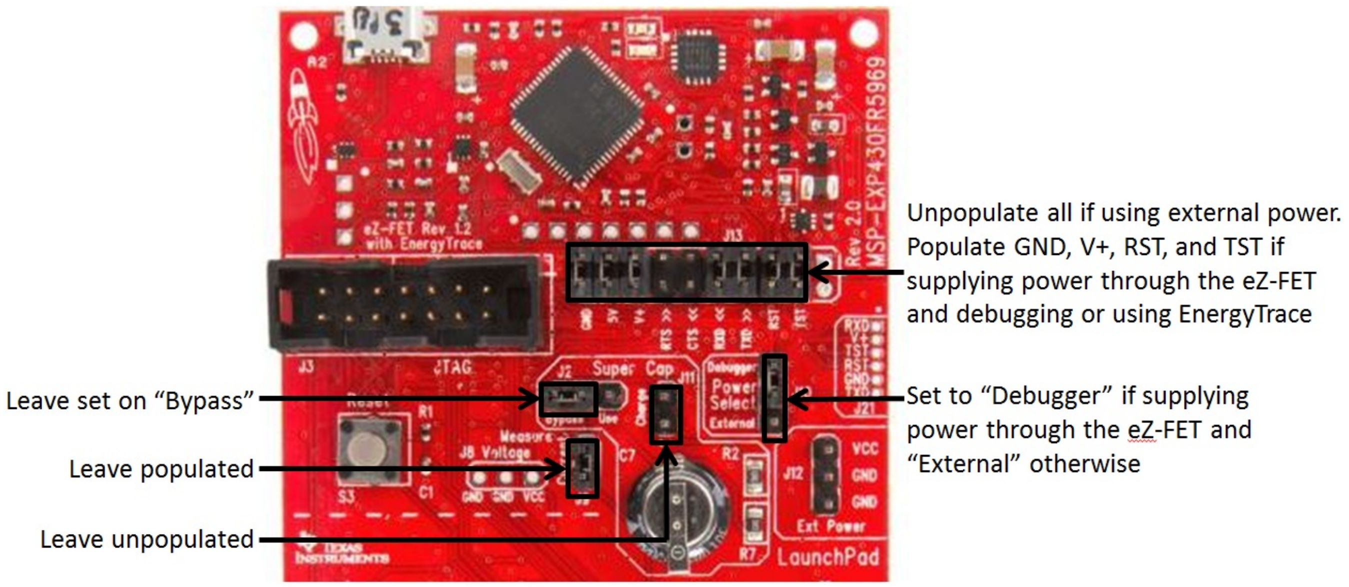

Unpopulate all if using external power. Populate GND, V+, RST, and TST if supplying power through the eZ-FET and debugging or using EnergyTrace
Leave set on "Bypass"
Set to "Debugger" if supplying power through the eZ-FET and "External" otherwise
Leave populated
Leave unpopulated

Figure 4. Orientation of MSP-EXP430FR5969 Headers

3.2 Parasitic Power Operation

All rules from the dedicated power section still apply during the parasitic power operation; however, the hardware schematic changes slightly to allow for the MSP430FR5969 device to harvest its power from the data line as compared to an independent power source. Connecting V_{cc} to the data line through the use of a Schottky diode, selected as such for a low dropout voltage, allows the device to avoid being directly connected to a power supply. A tank capacitor, C_{Tank}, is then connected from the V_{cc} to GND to store an additional charge while the data line is held low for communication purposes. The Schottky diode used in this application must have a forward voltage drop of less than 250 mV so that it does not violate the MSP430 device's voltage rating between the V_{cc} and a GPIO. Review Figure 2 for a firm understanding of the hardware changes that are needed to operate in parasitic power mode. Table 1 lists the recommended values and part numbers.

The value of R_{Pup} and C_{Tank} determine the RC time constant (time required to charge and discharge a capacitor), and may be altered slightly depending on the number of byte transfers desired in a worst-case scenario without draining C_{Tank} and causing the MSP430 device to enter a reset state. For more information on parasitic power operation, refer to Section 4.

4 Firmware Considerations

4.1 Code Overview

A software package is included in this TI Design that contains application firmware designed for the purpose of 1-Wire EEPROM emulation on an MSP430FR5969 device. To evaluate the TIDM-1WIREEEPROM firmware, the CCS project must be imported using CCS V6.0.1 (or later), and loaded onto an MSP-EXP430FR5969 board. Due to the small amount of peripheral restrictions contained within this design, the firmware may be easily reorganized for implementation onto any MSP430 FRAM device. The code may further be altered for operation of other 1-Wire applications such as RTC, battery monitoring, and temperature-sensing applications. Figure 5 shows the main block diagram of this firmware.

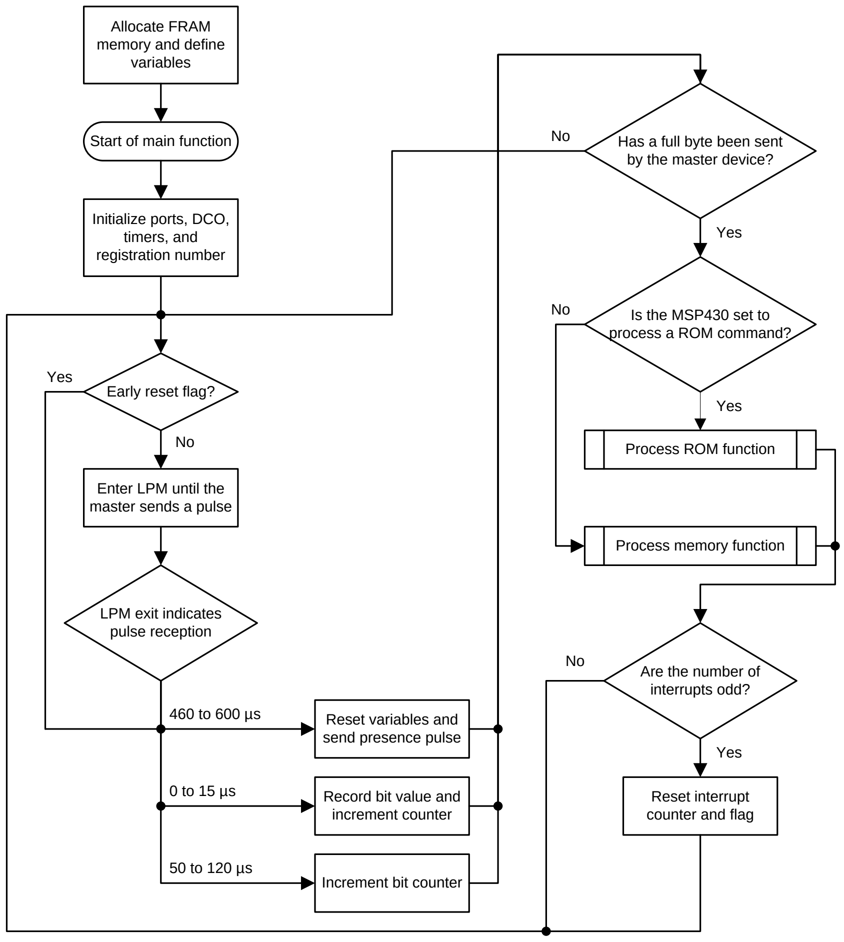

Allocate FRAM memory and define variables
Start of main function
Initialize ports, DCO, timers, and registration number
Yes
Early reset flag?
No
Enter LPM until the master sends a pulse
LPM exit indicates pulse reception
460 to 600 µs
Reset variables and send presence pulse
0 to 15 µs
Record bit value and increment counter
50 to 120 µs
Increment bit counter
No
Has a full byte been sent by the master device?
Yes
No
Is the MSP430 set to process a ROM command?
Yes
Process ROM function
Process memory function
No
Are the number of interrupts odd?
Yes
Reset interrupt counter and flag

Figure 5. Firmware Main Block Diagram

This code uses LPM4 during standby mode for low power consumption, while the master device is not requesting any correspondence. Waking from LPM4 indicates that communication has begun with the master sending a reset pulse, and the length is measured by using the CCI1A module. All pulses received after initialization may result in one of three options: another reset command, a bit of value zero, or a bit of value one. A reset command results in a presence response, followed by the resetting of the counter. Otherwise, the bit value is stored, the counter is incremented for the next bit until a complete byte is filled, and a command subfunction is entered. The device re-enters standby mode and the entire process is repeated. The following sections cover the flow of device operation, including the purpose of the registration number and address registers, the proper transaction sequence, how communication is initialized, and detailed information regarding ROM and memory function commands. Some design restrictions are also listed for user awareness.

4.2 Design Restrictions

Because of the sampling resolution required in adherence to the 1-Wire protocol, the MSP430 device must operate with a minimum processor frequency of 16 MHz. The firmware is optimized for this frequency and altering this would require multiple changes to the timing of the code. The MSP430 device must only enter LPM0 while the master is active as more productive LPMs do not have enough wake-up time to properly measure and respond to the shortest pulses being sent. It is specifically when a reset pulse that the device will enter LPM4.

NOTE: Overdrive mode cannot be realized with the 1-Wire solution on MSP430 devices and has not been included as an option in the firmware.

While operating in parasitic mode, a maximum number of worst-case bytes (being 00h because it holds the data line down for the longest amount of time), can be sent continuously before C_{Tank} is discharged. The MSP430 device does not have enough power to continue operation, and enters a reset state. The maximum value is dependent on both R_{Pup} and C_{Tank}, which define the RC time constant, and as the VCC level and pulldown strength of the master device. With a test setup involving a TM4C129XNCZAD development board, R_{Pup} has a value of 1 kΩ, and C_{Tank} has a value of 22 µF. The maximum number of worst-case transfers was about 100 bytes. This result could be further improved by lowering the value of R_{Pup} to less than 280 Ω and increasing the value of C_{Tank} as high as the design realistically allows. However, for most applications, this discrepancy will not allow sending of the complete EEPROM contents at a single time. Instead, the memory must be sent in predetermined sizes with a delay in-between to recharge C_{Tank}. Further guidance on this process includes a firmware solution that is provided in Section 4.8. The complete EEPROM contents may be sent continuously while in dedicated power mode.

If using an MSP430FR5969 device to emulate 1-Wire EEPROM functionality, such as in this design guide use of the MSP-EXP430FR5969, a unique operating condition that must be considered. If responding to a read memory command in parasitic power mode without properly considering the amount of EEPROM memory that may be sent during a single C_{tank} charge, it is possible that the MSP430 device may lower the power rail past the SVSH power-down level and cause a device reset. Once the device reset occurs, the MSP430 device no longer pulls on the shared data/V_{cc} line, allowing the power rail to return to its operational voltage. However, the master may prevent this from happening by issuing a reset pulse shortly after the MSP430 device is already in a reset state. The reset pulse may cause the MSP430 device to enter a BOR condition where Table 5-1 of Section 5.12 appears in the MSP430FR5969 Datasheet (SLAS704) requires that the device's V_{cc} to go all the way down to GND before powering up again. To alleviate this problem, the master must wait an additional amount of time (depending on the value of R_{pup} and C_{tank}) after a read memory command is completed before issuing a reset command. Otherwise, if a BOR condition has already been entered, then the master must hold down the shared data and V_{cc} line until the MSP430 device power rail reaches GND and then release so that the device may return to its normal operating voltage. This precaution is only valid if parasitic power mode while using a MSP430FR5969 device is desired.

4.3 Registration Number

During the initialization stage of the firmware, a 64-bit device-specific registration number is created for differentiating between and verifying 1-Wire slave devices connected to the data line. It is called upon by several of the ROM function commands explained in Section 4.6. The least significant byte (LSByte) is a family code that the user may determine by changing the value of the family constant. The following 6 bytes consist of the Lot or Wafer ID, and LSBytes of both the Die X and Y positions. These values are found in the Die Record section of the Device Descriptor (TLV) table in the MSP430FR5969 device memory, and are unique for each device. They remain as constant values during the lifetime of the device, and must not be replaced or substituted in the registration number under any circumstances. The most significant byte is a CRC code generated from loading the previous 7 bytes into a CRC8 algorithm.

4.4 Address Registers

Two address registers are used by the firmware: the 2-byte target address (TA) and the ending address with data status (E/S) byte. TA represents the address where data is to be written to or read from. The TA register is 2 bytes long, and may extend to a maximum address of 0xFFFF. This is limited by the amount of FRAM space allocated for EEPROM memory storage that is determined by the MEMORY_SIZE defined in the firmware. The firmware can be modified in accordance with the application requirements. The end address represents the offset from the target address when writing to the 32-bit-long scratchpad, and explains why it is only 5 bits (representing a maximum offset of 31) long.

The sixth and eighth bits of the E/S byte represent two different data status flags: authorization accepted (AA) and the partial byte flag (PF). AA indicates when data stored in the scratchpad is successfully copied to the target memory address, whereas a set PF reveals that data was not correctly written to the scratchpad during the last attempt. The seventh E/S bit is void and only reads a value of zero. More information regarding these flags is provided in Section 4.7.

4.5 Transaction Sequence

All communication with a 1-Wire EEPROM device must adhere to the following protocol:

1. Initialization
2. ROM function command
3. Memory function command
4. Transaction/data

The process is then repeated until the master no longer needs to send instructions to or require additional information from the slave device. The following sections explain in detail each step of the transaction sequence.

4.6 Initialization

Every transaction on the 1-Wire protocol begins with the initialization sequence, that consists of a reset pulse issued by the master, followed with a presence pulse delivered by the slave. This exchange of pulses synchronizes each side of the data line and prepares the pulses to begin a new transaction. The firmware is designed to recognize reset pulses from 460 to 600 µs long, and will respond (after a 30-µs delay) with a pulse 100 µs long. The firmware then enters the LPM0 to conserve power, but is still prepared to quickly acquire the next byte from the master that is expected to be a ROM function command.

4.7 ROM Function Commands

The purpose of this command is to ensure that the desired 1-Wire device is properly connected and ready to receive a memory function command from the master. This task is primarily accomplished through use of the unique registration number of the slave. This number helps the master differentiate between connected devices. Alternative commands have also been provided to skip identification if not required. Figure 6 shows the ROM function command block diagram.

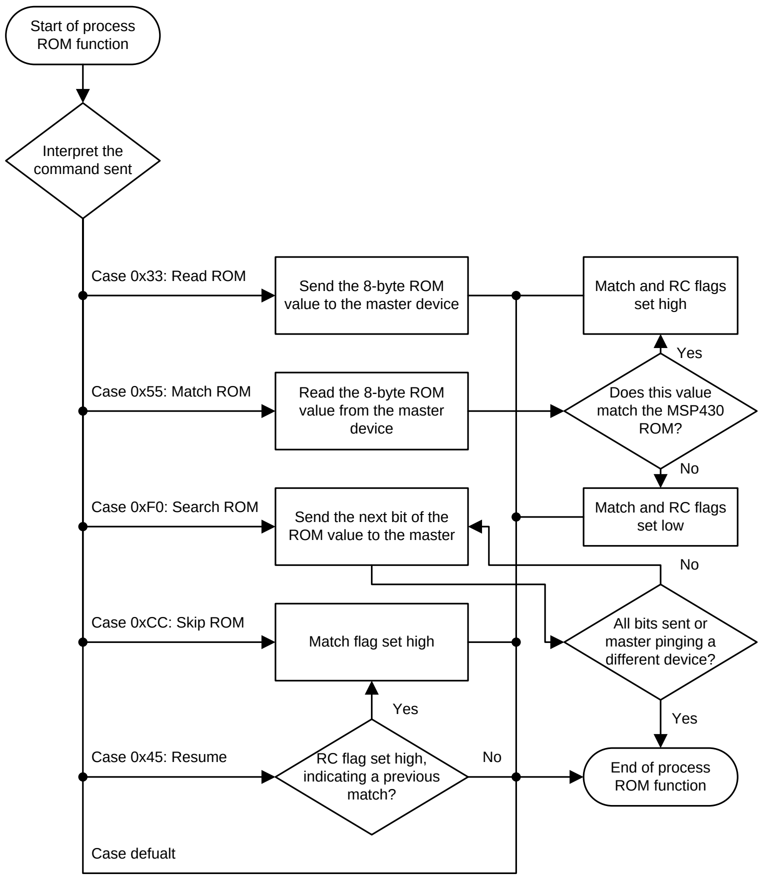

Start of process ROM function
Interpret the command sent
Case 0x33: Read ROM
Send the 8-byte ROM value to the master device
Match and RC flags set high
Yes
Case 0x55: Match ROM
Read the 8-byte ROM value from the master device
Does this value match the MSP430 ROM?
No
Case 0xF0: Search ROM
Send the next bit of the ROM value to the master
Match and RC flags set low
No
Case 0xCC: Skip ROM
Match flag set high
All bits sent or master pinging a different device?
Yes
Yes
Case 0x45: Resume
RC flag set high, indicating a previous match?
No
End of process ROM function
Case defualt

Figure 6. ROM Command Function Block Diagram

A description of each ROM command follows:

- Read ROM (0x33): The MSP430 device responds by sending the 64-bit registration number to the master device. This command can be used only if there is a single slave on the bus, otherwise data collision will occur.
- Match ROM (0x55): The master follows this command with a 64-bit registration number and only the slave device that precisely matches this ROM sequence responds to the memory function command that follows. The RC flag is also set high for purposes associated with the Resume command. All other slave devices on the bus hold their respective RC flag low and wait for a reset pulse.
- Resume (0xA5): Only the slave device that was most recently connected to the master through the Match ROM command (that is, its RC flag is set high) responds to the subsequent memory function command; the remaining slaves once again remain dormant until a reset occurs.
- Skip ROM (0xCC): If slave identification is not required on a single-bus system, this command allows access to memory function without providing the 64-bit ROM code. This command cannot be used with multiple slaves on the bus because this causes transmission confusion.
- Search ROM (0xF0): For use when there are multiple slave devices on the bus, this command helps the master differentiate between the registration number of each slave device by applying a process of elimination. To start, each slave device sends the least significant bit (LSB) of the ROM of that slave device followed by the complement of that bit. The master then responds with a bit value. If the bit sent by the master matches the bit value of the slave, then the process continues until the entire 64-bit registration number has been sent. However, if at any point the bit values do not match, then the slave device ceases operation until a reset pulse transpires. The master will likely repeat this command until all slave devices on the bus have been identified.

Assuming that the MSP430 1-Wire-compatible device is requested, the match flag is set to indicate that the next byte received by the master must be a memory function command after which a transaction of information occurs. If the byte does not correctly translate to a ROM function command, then the default action is to have the MSP430 device continue reading bytes until it either receives a proper ROM command or receives a reset pulse. As explained in Section 4.6, the firmware uses LPM0 while waiting for a memory command byte so that it may efficiently conserve power.

4.8 Memory Function Commands

Now that the correct 1-Wire-compatible device has been correctly identified and addressed, legitimate EEPROM functionality may begin. This will either come in the form of modifying the scratchpad contents, or reading from or writing to the device memory. Before the command is executed, the match flag must be reset so that the next byte processed will be a ROM function command during a new transaction sequence. If a nonsensical memory command is sent by the master device, the next byte sent must either be a reset pulse or repeated ROM command. Figure 7 shows the memory function command block diagram.

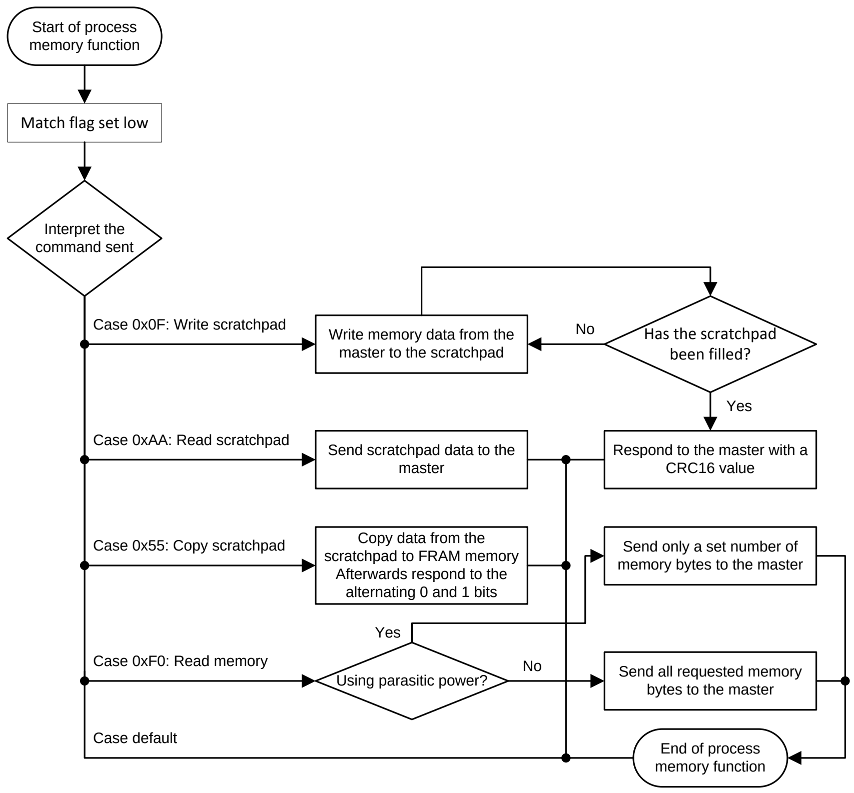

Start of process memory function
Match flag set low
Interpret the command sent
Case 0x0F: Write scratchpad
Write memory data from the master to the scratchpad
No
Has the scratchpad been filled?
Yes
Case 0xAA: Read scratchpad
Send scratchpad data to the master
Respond to the master with a CRC16 value
Case 0x55: Copy scratchpad
Copy data from the scratchpad to FRAM memory Afterwards respond to the alternating 0 and 1 bits
Send only a set number of memory bytes to the master
Yes
Case 0xF0: Read memory
Using parasitic power?
No
Send all requested memory bytes to the master
Case default
End of process memory function

Figure 7. Memory Command Function Block Diagram

Each memory command is explained below:

- Write Scratchpad (0×0F): The master immediately follows this command with the 2-byte target address and the data that is to be written to the scratchpad. Data being written to the scratchpad starts at the offset indicated by the 5 LSBs of the TA, the same value that is initially loaded to the ending address bits of the E/S byte. The value of E/S is then incremented with each subsequent byte written to the scratchpad. When the scratchpad is filled, the MSP430 device performs a 16-bit CRC algorithm (starting with the command code and ending at the last data byte) and returns the inverse of the CRC16 value to the master. When performing the Read Scratchpad command, the master can then compare this CRC16 to its own value calculated using the bytes received from the slave device. This is all done to verify the scratchpad contents before writing to the device memory through the Copy Scratchpad command. The master device can optionally choose to end the Write Scratchpad command at any time and not completely fill the scratchpad, in which case CRC16 functionality is not performed by the slave device. The partial flag (PF) bit is set high at the beginning of this memory command and is only reset after writing to the scratchpad completes, therefore indicating to the user if communication was interrupted during the process.
- Copy Scratchpad (0×55): This command is used to copy data from the scratchpad to FRAM memory. The master must first send the proper TA and E/S address registers which can be obtained through a preceding Read Scratchpad command. If this is successful then the AA flag of the E/S register is set and the device begins writing data. With this format it can be imagined that the EEPROM storage is divided into 32-byte sections of the same length as the Scratchpad, and only one section can be

written to with each copy command. Once completed then the MSP430 device responds to the master with alternating bit values of 1 and 0 and continues to do so until a reset pulse is detected. If in parasitic power mode then the alternating bit pattern eventually drains C_{Tank} resulting in a device reset thus ending the reply; however, this will not affect the data already successfully written to the FRAM memory.

- Read Scratchpad (0×AA): The slave responds by sending the 2-byte TA, E/S byte, and the scratchpad contents beginning at the offset determined by the 5 LSBs of the target address. This helps verify both of the address registers and scratchpad data.
- Read Memory (0×F0): This is the general command used to read the MSP430 FRAM memory allocated for EEPROM data storage. The master must supersede this command with the desired 2-byte target address within the defined memory size range. If operating in dedicated power mode, the slave replies with all data bytes starting with the target address and continuing until the end of the designated memory is reached. But in parasitic power mode the values USE_PARASITIC_POWER and PARASITE_BYTES should be defined by the firmware so that, starting at the target address, the MSP430 device responds only with the number of bytes set by PARASITE_BYTES. In doing so, the device can ensure that a predetermined segment of memory is sent before C_{Tank} is depleted and the device enters a reset state. The master should then briefly pause communication to allow CTank to fully charge again before repeating the process, this time offsetting the target address from the bytes already accessed, to finish reading the desired FRAM memory.

After receiving the memory command and processing the transaction/data accordingly, the device re-enters LPM0 (with the exception of the copy scratchpad command) from which the transaction sequence is allowed to restart whenever desired. If inactivity lasts longer than a millisecond, the device initiates standby mode by entering LPM4 so that power consumption is further reduced. But in the case of a copy Scratchpad command, the MSP430 device must signal completion by sending alternating bit values instead of entering any LPMs. It will continue to do so until the master issues a reset pulse or, in the case of parasitic power mode, until C_{Tank} completely discharges and the MSP430 device is forced to enter a reset state. In this case, the device would no longer continue to pull on the data line, allowing C_{Tank} to recharge and reboot the MSP430 device. At this point, the device is prepared to start another transaction sequence.

5 Test Data

5.1 Transaction Sequences Example

To further aid in the production of the logic analyzer screen shots and EnergyTrace recordings provided in the following sections, an example series of transaction sequences was generated. In this instance, there are five individual sequences whose purposes are to:

- Identify the slave device using the Search ROM command.
- Write to the Scratchpad.
- Read and verify the Scratchpad contents.
- Copy the Scratchpad to the allocated FRAM memory.
- Read back the entirety of the EEPROM memory.

The Skip ROM command was included to prove applicable operation, but only one slave device must be connected to the bus. Figure 8 shows the explanation pertaining to the application of these sequences.

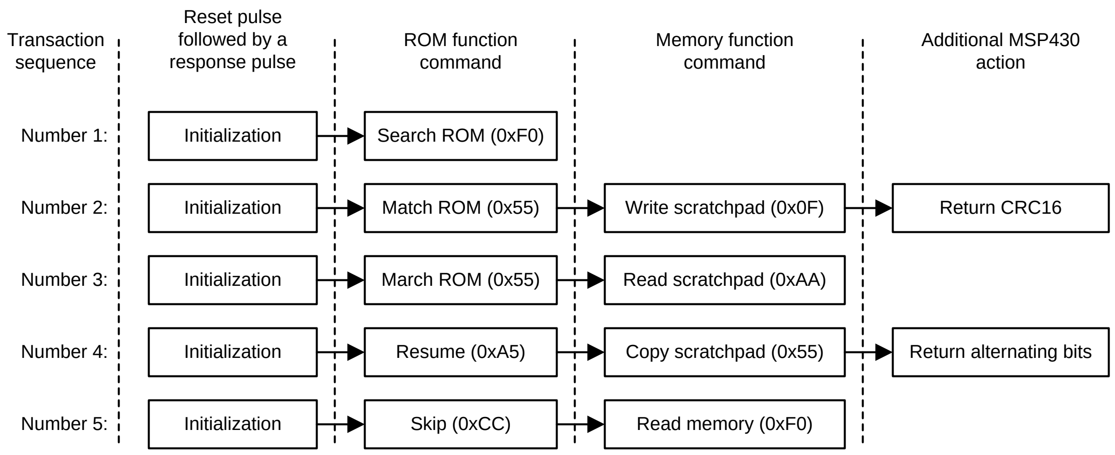
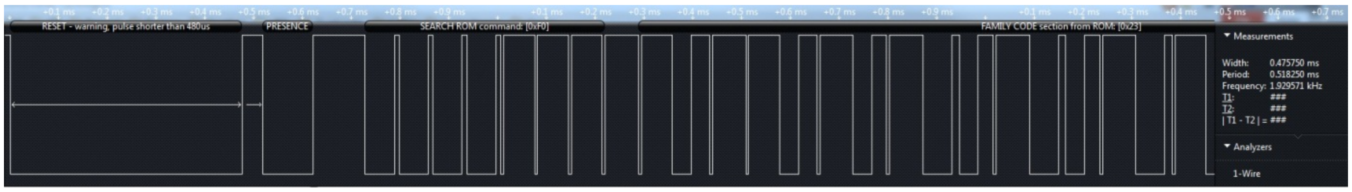


Figure 8. Description of Transaction Sequences Example

5.2 Logic Analyzer Screen Shots

A logic analyzer was used to test and confirm that the MSP430FR5969 device's operation was compatible with the 1-Wire protocol. This setup was also effective in comparing the MSP430 device solution to other 1-Wire EEPROM devices to demonstrate identical communication and functionality. Several screen shots have been provided to validate the operation and effectiveness of the design with common 1-Wire ROM and memory commands.

Figure 9 shows the initiation of a transaction sequence beginning with a reset pulse sent by the master, and responded to with a presence pulse from the MSP430 device. By looking at the measurements tab of the logic analyzer environment, the reset pulse is approximately 480 µs long. The MSP430 device waits a further 40 µs before replying with a presence pulse of about 120 µs. The master then recognizes the existence of a slave device on the bus, and continues by transmitting a valid ROM function command.


+0.1 ms +0.2 ms +0.3 ms +0.4 ms +0.5 ms +0.6 ms +0.7 ms +0.8 ms +0.9 ms
RESET - warning, pulse shorter than 480us
PRESENCE
SEARCH ROM command: [0xF0]
FAMILY CODE section from ROM: [0x23]
Measurements
Analyzers
1-Wire

```
Width:      0.475750 ms
Period:     0.518250 ms
Frequency:  1.929571 kHz
T1:         ###
T2:         ###
| T1 - T2 | = ###
```

Figure 9. Presence Pulse Example

Figure 10 shows the end of a write scratchpad command, where 0×02 is the last data byte sent by the master needed to fill the scratchpad. The MSP430 device now employs the CRC16 algorithm on all bytes sent by the master, starting with the memory command code, and ending at the last data byte. Then return the inversed CRC16 value to the master device as shown in Figure 10 by the 0×EA and 0×9C bytes. The values output by the MSP430 device firmware have been cross checked with the CRC16 algorithm and other 1-Wire devices to assure proper operation. This functionality is optional and can be avoided by either not completely filling the scratchpad or not sending any read time slots after all data bytes have been sent.

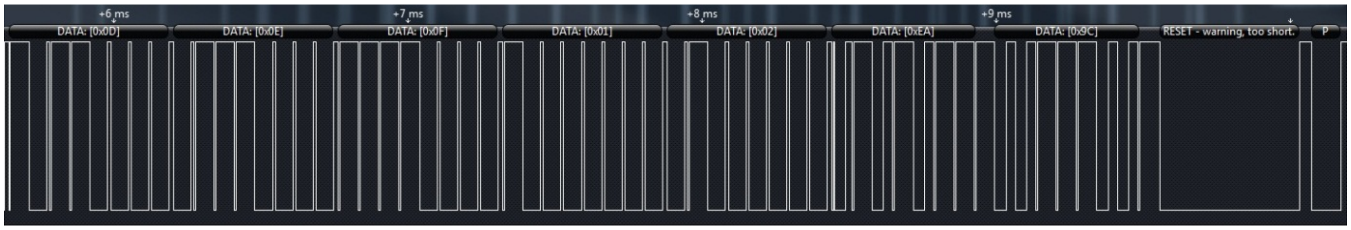


Figure 10. Write Scratchpad Example With CRC16 Response

Figure 11 shows a copy scratchpad command where the master provides the correct 3-byte authorization pattern. This process allows the MSP430 device to begin copying the contents of the scratchpad into the allotted FRAM memory. Copying the contents only takes a few microseconds, and afterward, the MSP430 device responds to the master with alternating 0 and 1 bits to signal completion of the command, shown in the 0×55 byte in Figure 11. While reading this response, the master issues a reset pulse to begin the next transaction sequence.

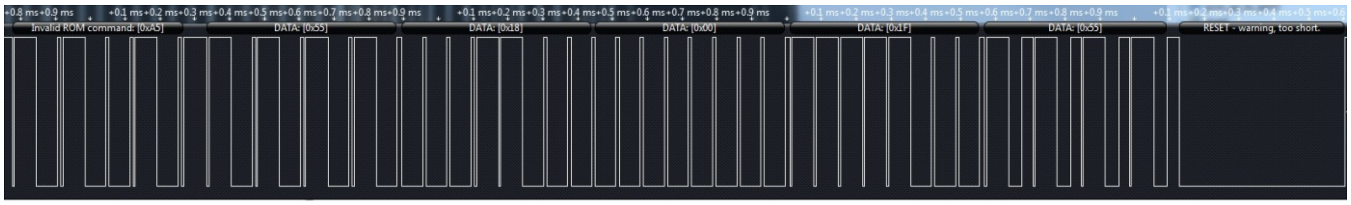


Figure 11. Copy Scratchpad Example With an Alternating Bit Response

Although lacking some context, Figure 12 shows the FRAM retention after a device reset. Figure 12 was recorded during a read memory command where one section of the EEPROM memory was copied from the scratchpad, the MSP430 device was power cycled, and then a different section of memory was written. Retention of the first segment of data is shown on the left side of the screen shot, and the newly recorded data is displayed on the right side with untouched bytes of memory separating the two. Because the two separate copy scratchpad commands were executed on opposite ends of a power cycle, the memory has been properly retained in the case of MSP430 device experiences power down or reset.

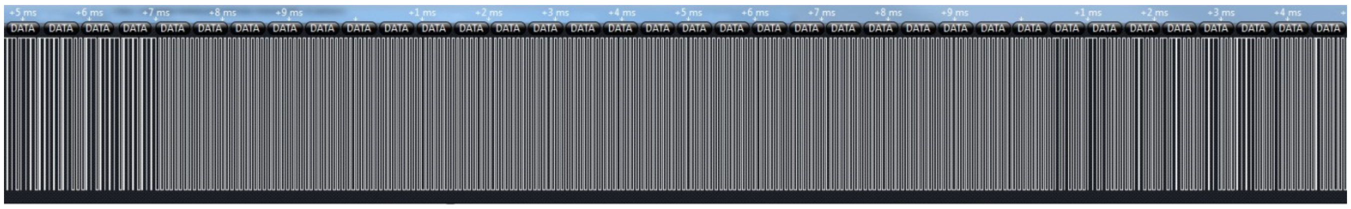


Figure 12. Memory Retention Example

The MSP430 device enters a reset state if trying to send a large array of bytes continuously while in parasitic power mode. The discharge of C_{Tank} that occurs while the data line is used for communication instead of holding the line high to power the MSP430 device. This is shown in Figure 13, where after repeating the memory contents of 0×00 continuously, the device eventually powers off. It will therefore no longer respond to the master read time slots that generate the 0×FF values. The data line must be allowed to stay high for C_{Tank} to recharge, where the transaction sequence can be restarted.

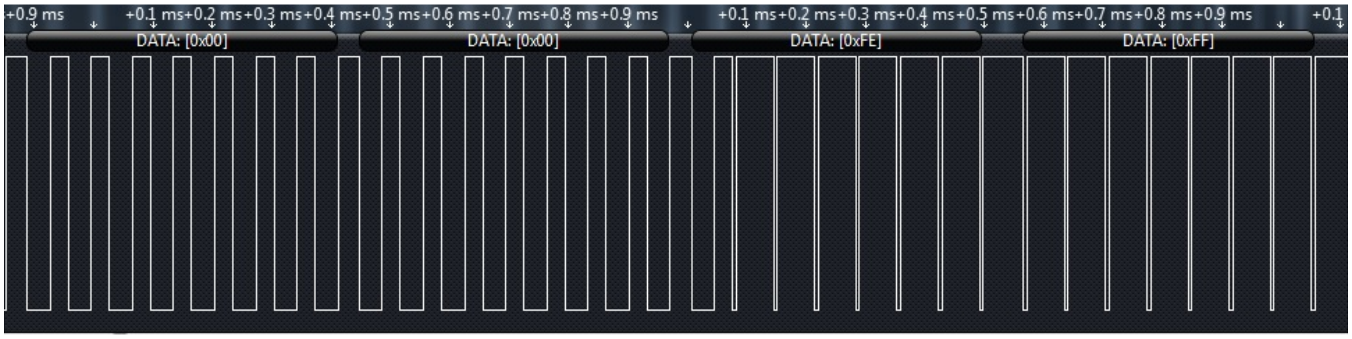
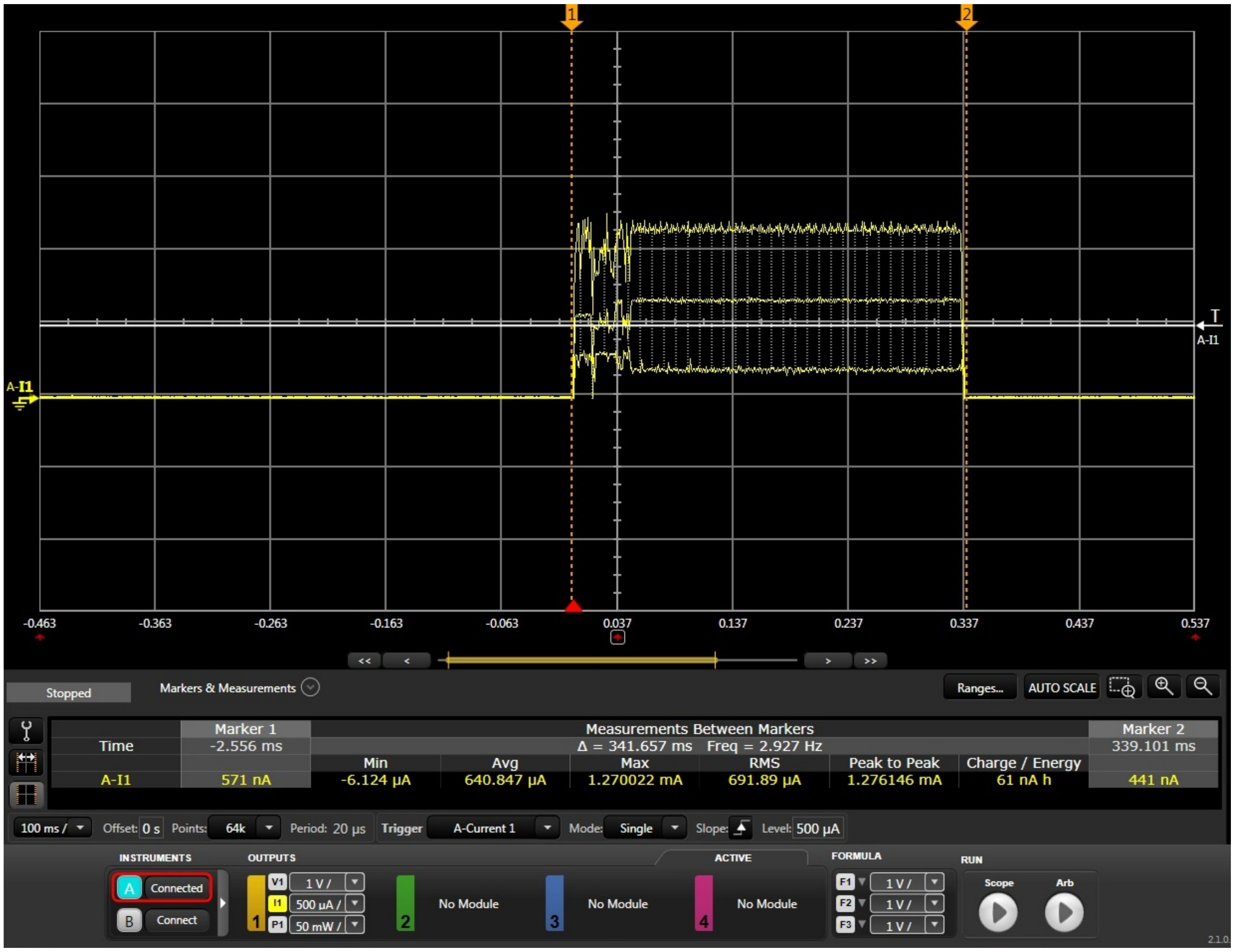

+0.9 ms
+0.1 ms+0.2 ms+0.3 ms+0.4 ms+0.5 ms+0.6 ms+0.7 ms+0.8 ms+0.9 ms
+0.1 ms+0.2 ms+0.3 ms+0.4 ms+0.5 ms+0.6 ms+0.7 ms+0.8 ms+0.9 ms
+0.1
DATA: [0x00]
DATA: [0x00]
DATA: [0xFE]
DATA: [0xFF]

Figure 13. Parasitic Power Operation C_{Tank} Drain Example

5.3 *Keysight EnergyTrace Recordings*

To further understand the energy requirements of the MSP430 device, power consumption statics for both active and standby modes were measured using a Keysight N6705B DC power analyzer. EnergyTrace is an energy-based core analysis tool that measures and displays the energy profile of the application and helps to optimize it for ultra-low-power consumption.

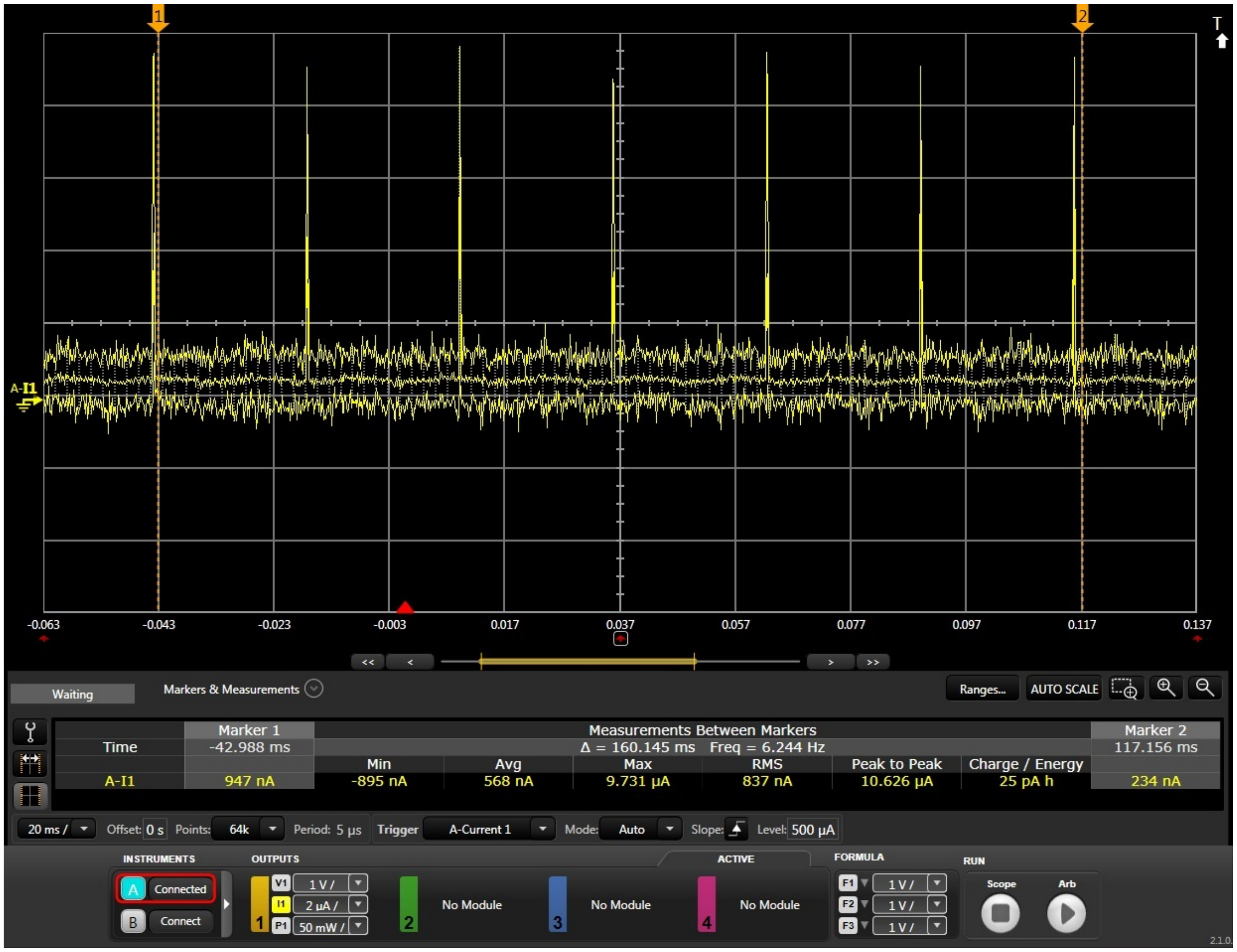
Figure 14 and Figure 15 show the current consumption measurements of active and standby mode, respectively, of the MSP430 device during the transaction sequences example as explained in Section 5.1. As shown in Figure 14, this entire process takes nearly 340 ms to complete in which an average of 640 µA of current is required for active mode as compared to the 570 nA consumed in standby mode, shown in Figure 15. During standby mode, the average current and displayed waveform, for which an immediate spike is shown every 26.6 ms to indicate SVS operation, closely reflects the expected LPM4 current numbers referenced in Section 5.7 of the MSP430FR5969 Datasheet (SLAS704).



	Marker 1	Measurements Between Markers						Marker 2
Time	-2.556 ms	Δ = 341.657 ms Freq = 2.927 Hz						339.101 ms
		Min	Avg	Max	RMS	Peak to Peak	Charge / Energy	
A-I1	571 nA	-6.124 µA	640.847 µA	1.270022 mA	691.89 µA	1.276146 mA	61 nA h	441 nA

100 ms / Offset: 0 s Points: 64k Period: 20 µs Trigger A-Current 1 Mode: Single Slope: Level: 500 µA

Figure 14. Power (in Milliwats) During Active 1-Wire Operation



	Marker 1	Measurements Between Markers						Marker 2
Time	-42.988 ms	Δ = 160.145 ms Freq = 6.244 Hz						117.156 ms
		Min	Avg	Max	RMS	Peak to Peak	Charge / Energy	
A-I1	947 nA	-895 nA	568 nA	9.731 µA	837 nA	10.626 µA	25 pA h	234 nA

20 ms / Offset: 0 s Points: 64k Period: 5 µs Trigger A-Current 1 Mode: Auto Slope: Level: 500 µA

Figure 15. Energy (in Microjoules) During Active 1-Wire Operation

Because the MSP-EXP430FR5969 ez-FET contains built-in EnergyTrace++ support, it is also possible to include information about the internal state of the microcontroller. These states include the on/off status of the peripherals as well as all system clocks and LPMs currently in use. This tool provides a means of directly verifying whether an application is demonstrating the expected behavior at the correct points in the code, such as ensuring that a peripheral is turned off after a certain activity or that the desired LPM has been entered properly. A full guide on EnergyTrace++ technology may be found in chapter 3 of the *Code Composer Studio v6.1 for MSP430 User's Guide* (SLAU157). Because EnergyTrace++ requires that the MSP430 device power be supplied by a software-controlled DC-DC converter found on the ez-FET portion of the MSP-EXP430FR5969 LaunchPad, the following readings were taken with the device in dedicated power mode.

Figure 16 shows the MSP430 1-Wire functionality during its active mode. The only peripherals used for this design include FRAM for storing EEPROM memory, TA0 to measure pulse widths, and TA1 used for general delays and other time measurements. Figure 16 shows that the majority of the active cycle is spent in LPM0 and that the device will later enter LPM4 when it is confident that the master is no longer requesting communication. The design has therefore optimized its ultra-low-power consumption capabilities. Figure 17 shows the schematic of the TIDM-1WIREEEPROM.

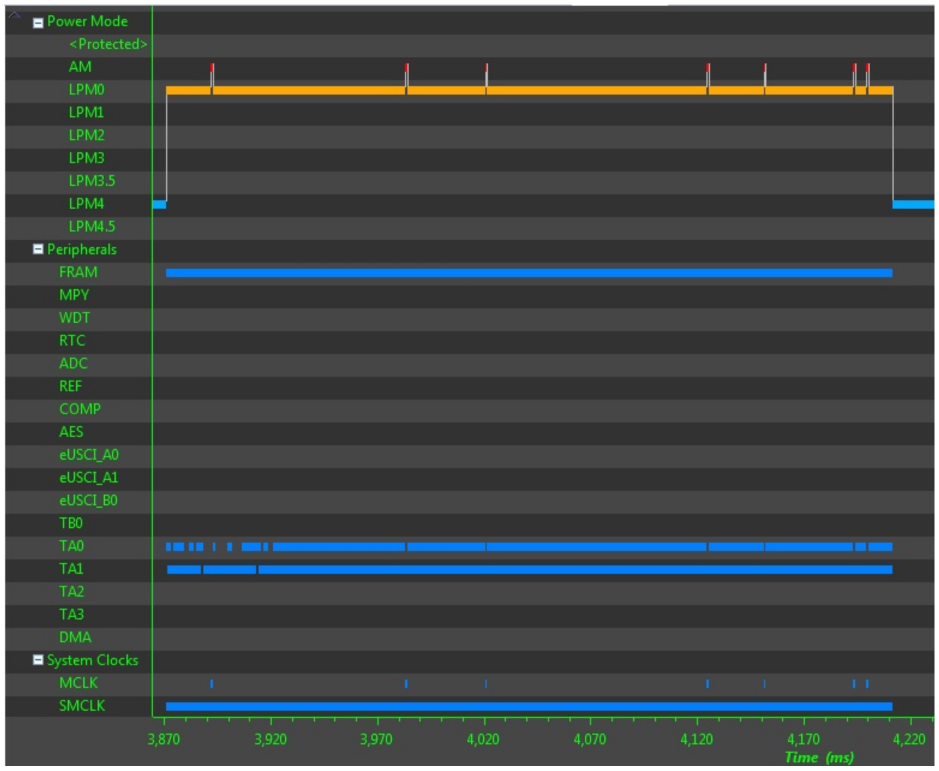


Figure 16. CPU and Peripheral States During Active 1-Wire Operation

6 Design Files

6.1 Schematics

To download the schematics for each board, see the design files at http://www.ti.com/tool/TIDM-1WIREEEPROM

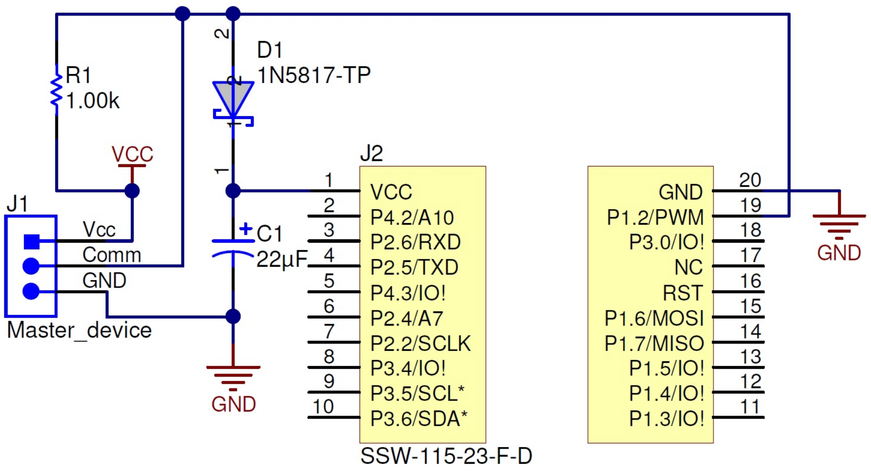


Figure 17. TIDM-1WIREEEPROM Schematic

6.2 Bill of Materials

To download the Bill of Materials for each board, see the design files at http://www.ti.com/tool/TIDM-1WIREEEPROM

Table 1. TIDM-1WIREEEPROM Bill of Materials

ITEM	QUANTITY	REFERENCE/ DESIGNATOR	VALUE	PART DESCRIPTION	MANUFACTURER	MANUFACTURER PART NUMBER	ALTERNATE PART	ALTIUM FOOTPRINT
1	1	C1	22 µF	CAPACITOR, AL, 22 µF, 25 V, ±20%, TH	Panasonic™	20SEP22M		M:_500×500
2	1	D1		Diode, Schottky, 800 mA, 20 V, SM	Toshiba™	CUS08F30, H3F		SOD-323
3	1	J1		Header, 100 mil, 3×1, Tin, TH	Wurth Electronics®	61300311121		5-146278-3
4	1	J2		Header, 100 mil, 15×1, Tin, TH	Samtec™	SSW-115-23-F-S		Custom
5	1	R1	1.00 k	RES, 1.00 kΩ, 1%, 0.25 W, TH	Stackpole Electronics™	CF14JT1K00		CMF50

6.3 Parasitic Power Mode Layout Recommendations

Because this design focuses on the MSP430 device firmware and does not require several additions to the hardware for operation, a PCB layout has not been developed. However, the user may wish to connect their MSP-EXP430FR5969 LaunchPad to the master device using parasitic power mode as described in Section 3.1, and shown in Figure 1. For design evaluation, TI recommends using a female header similar to that of the Samtec SSW-115-23-F-S.

This 15-position elongated-lead header is placed across the right-side headers of the MSP-EXP430FR5969 and extra V_{CC}/GND headers at the bottom right side of the board. R_{Pup} then connects from P1.2 (where the CCI1A/data line input is located) to an unpopulated header. In turn, the unpopulated header us connected to the DSchottky's anode end of the cathode end connects to the V_{CC} header. C_{Tank} is connected across the V_{CC} and G_{ND} headers. Use standard jumper cables, the master V_{CC} is then connected to the R_{Pup}/$D_{Schottky}$ unpopulated header, data line to P1.2, and GND to GND. Table 1 lists the recommended hardware part numbers.

6.4 MSP430 FRAM Device Porting Advisement

If it is desired that a FRAM MSP430 device other than the MSP430FR5969 device be used for 1-Wire EEPROM emulation, there are a few peripheral porting guidelines that must be considered. First, the replacement MSP430 device must be capable of operating at a CPU speed of 16 MHz. The device should also have at least one CCIxA/B interrupt-capable pin and a separate timer for general clocking purposes. The initialization of both the clock system and timers in the TI Design's firmware must be re-evaluated accordingly. Regarding the FR2xx/FR4xx family, the MPU code must be replaced with FRAM write protection, located in the SYSCFG0 register. More information on FRAM write protection may be found in Section 1 of the *MSP430FR4xx and MSP430FR2xx Family User's Guide* (SLAU445). A CCS v6.1 project for 1-Wire EEPROM operation using a MSP430FR4133 device has already been encompassed in the software package included with this TI Design.

7 Software Files

To download the software files for this reference design, see the link at http://www.ti.com/tool/TIDM-1WIREEEPROM.

8 References

1. *1-Wire*, https://en.wikipedia.org/wiki/1-Wire
2. *1-Wire Enumeration* (SPMA057).

9 About the Author

RYAN M. BROWN is an MSP430 device Customer Applications Engineer at TI, where he is responsible for supporting development of customer designs in various applications. Ryan brings to this role his experience in PCB design software, MSP430 architecture, and LaunchPad and BoosterPack protocol. Ryan earned his Master of Science in Electrical Engineering (MSEE) from Texas Tech University in Lubbock, TX.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (June 2016) to A Revision	Page
• Changed software link to correct location	21